Broadview®
www.broadview.com.cn

无线电安全攻防大揭秘

360独角兽安全团队(UnicornTeam) 杨卿 黄琳 等著



中国工信出版集团 電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

VISIT...

LANZAROTE Caliente.COM

版权信息

书名：无线电安全全攻防大揭秘
作者：杨卿等
出版社：电子工业出版社
ISBN：9787121285806
定价：69.00

版权所有·侵权必究

推荐序1

本人已从事通信理论的研究与教学工作二十多年，曾承担多项国家级科研项目以及与企业合作开发项目，在通信系统设计、算法优化和协议实现等方面积累了一些经验。

应该讲，所有通信系统都存在安全问题，通信系统安全问题涉及设备安全、内容安全和防护安全等，各种安全措施伴随着通信和网络技术的发展而发展，特别是当今互联网技术的发展引发的安全问题越来越突出。

近年来，国家层面对网络空间安全越来越重视，无线电的安全问题也越来越受到政府、企业和公众的关注。伪基站、非法广播电台，已经真实地影响到普通老百姓的生活。并且，随着软件无线电技术的发展，无线通信协议的实现成本变得越来越低，这意味着攻击的门槛也变得越来越低。在2009年，我的学生在研究OpenBTS的时候，就已经发现搭建的基站很容易把周围的手机吸入它的网络，这也许是国内最早的伪基站。只是那时，我们从未想到，伪基站会泛滥成如今的样子。

本书作者黄琳是我的学生，一位具有丰富经验的无线通信领域的专家，对无线通信和无线电安全进行了长期深入研究和实践。2010年，她曾写过一本《GNU Radio入门》教程，在网络上影响很大。她还曾作为中国360UnicornTeam的唯一女性代表在第23届DEFCON世界黑客大会上发表演讲，在无线电安全领域产生了重要影响。

《无线电安全攻防大揭秘》这本书，涵盖当前常见无线电多种应用的安全问题，内容讲解深入浅出、通俗易懂、可读性强，具备一般专业基础知识的读者均可阅读。本书既适合从事无线电安全领域的技术人员阅读，对无线电安全方面感兴趣的一般读者也可以通过本书对该领域有初步的了解。

王文博（北京邮电大学教授，研究生院常务副院长）

推荐序2

无线电安全是一个既年轻又古老的领域。有多古老呢？和无线电本身一样古老。

1903年，当马可尼在伦敦皇家学院公开展示无线电通信时，他的竞争对手马斯基林就劫持了马可尼演示的通信。马斯基林在发射的信号中骂马可尼是“老鼠”、“欺骗公众的意大利佬”，弄得现场非常尴尬。直到今天，信号劫持及其防御，仍然是无线电安全领域的重要内容。

二战期间，盟军成功破解纳粹用Enigma机加密的无线电报，大大改变了战局。盟军和纳粹当年在加密解密对抗上所做的工作，今天也仍是无线电安全领域的重要内容。

随着时间的推移，无线电通信技术日趋先进、便捷和普及，无线电安全领域的攻防对抗也逐渐从神秘的实验室，走到了大众的身边。20世纪80年代，使用模拟信号的移动电话“大哥大”在香港流行，市面上也就随之出现了专门用于窃听“大哥大”的“小弟小”（粤语叫“细佬细”）。

进入21世纪后，信息技术一日千里，作为其重要组成部分的数字无线电通信技术，也以惊人的速度在发展。然而，人们对这个领域安全对抗的了解却一度没有跟上。十几年前，黑市上刚出现GSM窃听装置时，还有不少通信领域的专家出来辟谣，认为GSM不可能被窃听。现在，GSM的不安全，至少在信息安全工作者中已经成为常识。

今天，在智能手机的引领下，万物互联的趋势不可阻挡，也带动了各种无线通信技术的发展。当街边的大爷大妈也能随口说出“WiFi”、“GPS”这些词的时候，当家门口的电表也被接上天线的时候，无线电安全越来越多被大众关注也就不奇怪了。然而，相对软件安全领域，无线电安全人才还是比较缺乏的。

我国每年培养的通信人才并不少，高校的安全专业建设这几年也

逐渐跟上来了。但懂通信、会逆向，能拆能焊，又有攻防思维的跨界人才就比较稀罕了。而未来，这个领域会越来越重要。希望这本书能帮助更多对此有兴趣的年轻人走上无线电安全研究之路。

无线传千里，攻防永不眠。

于旸（腾讯玄武实验室总监，网名“tombkeeper”）

推荐序3

I first meet Yang Qing at Blackhat2014and was pleasantly surprised to learn such ayoung face leads ateam on hardware security.Since then he has accomplished several works，ranging from GPS spoofing to designing various gadgets for protecting users’security and privacy.When he told me about their book on wireless security，I was delighted because wireless security is such an important topic and needs much more attention and talents than what we have today.

With the proliferation of wireless technologies，numerous emerging wireless devices have been woven into the fabric of our daily life，ranging from controlling our home appliances to making our vehicles automatically seek for help in emergency situations.Unfortunately，the security on these wireless devices has almost always lagged behind the plethora of the interests in integrating wireless technologies into almost everything.Granted that manufactories and designers have gradually increased their motivation in securing their devices，we have along way to go.Spreading knowledge on wireless security is one of the critical efforts towards securing wireless devices，and this book serves as agood endeavor along this goal.

Many academic books on wireless communication and wireless security are available，and many of them focus on the theoretical principles.This book is acollection of the systems works that Qing and his team have carried out in the past few years as well as the state of the art.The book covers awide range of wireless devices，such as RFID，Bluetooth，Zigbee，GPS，and it contains many results，plots，screen-snapshots from real world experiments.

This book can serve as agood tutorial for those who want to

have their hand dirty and reproduce the prior findings.It can also be agood reference book for those who want to find out the off-the-shelf tools for exploring the wireless world.I hope this book can help to foster talents in wireless security and to teach them necessary skills to secure the wireless world for many years to come.

徐文渊（浙江大学教授，南卡大学终身教授）

前言

本书的由来

想到在广泛的无线频谱里，有那么多技术应用可以被我们研究，寻找漏洞，并通过漏洞影响一个又一个现实的设备，如手机、电脑、汽车、传感器、工控机、智能家居，甚至是植入在人体内的各种医疗设备，你会不会感到很兴奋呢？是的，这其实就是我及我的团队一直专注的方向和领域。无线电技术是人类能前进到此刻的重大发明之一，我们不能忽视这里面可能发生的安全问题，我期待在未来会有更多的朋友加入一同在这一安全领域遨游。我们团队在国内还将继续作为先驱者带动这个安全领域的健康发展。这也是我们将多年来的所有积累编著成书的原动力。

同时希望本书也可以为对无线通信安全感兴趣的同学、从业者、产品研发人员提供有价值的安全参考。

本书的结构

第1章介绍我本人在无线安全攻防领域多年来的思路、理念以及对该领域未来的展望。

第2~8章分别介绍无线通信主流技术的安全攻防（RFID、射频、ADS-B、BLE、ZigBee、移动通信网络、卫星通信等）及实例测试。由独角兽安全团队（UnicornTeam）核心成员黄琳、单好奇、张婉桥、李均共同编写。

第9章作为无线安全研究实操的核心内容，介绍无线安全研究工

具GNU Radio的详细使用。本章由《GNU Radio V0.99入门》作者黄琳基于旧版本更新而来，是不可多得的珍贵教程。

致谢

感谢360公司周总、齐总为独角兽团队全体成员提供了自由开心的工作环境与富有竞争力的福利待遇。

感谢360公司首席安全官谭晓生对独角兽团队成员工作的指导与栽培。

感谢360信息安全部门0keeTeam张鲁、VulpeckerTeam宋申雷、NirvanTeam高雪峰在工作上的支持与帮助。

感谢360人力资源部王文萍、梁丽丽、王娜在工作上的支持与帮助。

感谢360公司市场传播部韩笑、尹乃潇、徐粲然、张震宇、马利娜、许传朝、陈晨、苑一时、景义哲、裴智勇、孙红娜等同事对我们团队工作的支持与帮助。

感谢360公司李洪亮、张卓、李向东、张睿、郑文斌、张聪、蔡玉光、唐青昊、林伟、刘小雄、赵晋龙等同事在工作上的协助。

感谢360独角兽团队黄琳、单好奇、张婉桥、李均、简云定、柴坤哲、郑玉伟、曾颖涛、袁舰、秦明闯、张强、张笔、王佳、郭怡婷、顾为群、徐佳、杨芸菲、王永涛、任兴典、张晓东、唐志红、崔婷等同事的努力付出。

杨卿（360独角兽安全团队总监）

第1章　鸟瞰无线安全攻防

1.1　无线安全概述

1.1.1　无线安全的由来

“无线”安全是庞大的信息安全体系下一门很广泛的学科。当今社会，电子产品大量依靠各种无线技术，从近场通信NFC、蓝牙BLE、射频RF、工控无线传输ZigBee、无线局域网WiFi，到手机蜂窝网络Cellular、卫星定位GPS、卫星通信SATCOM，这些都属于无线技术领域，所以无线通信技术上的传输、认证、加密等安全问题，在各种设备对无线技术依赖加深的情况下变得越来越重要。从现在到未来，人类对这些技术的安全拥有足够的掌控也将是非常必要的。因此，如何正确地使用无线通信技术，并保证其安全，是每一个研发、产品、安全研究人员都要认真思考的问题。

遥想10年前，国内安全圈里的黑客们还在最早的无线安全时代（无线局域网、WiFi）研究与徘徊，当时破解WiFi密码、蹭网继而进行网络渗透是最传统的无线攻击方法。那个时代，也是无线局域网安全最火热的年代，无线安全研究者们专注于寻找性能优良的无线网卡，搭配自己优化的无线破解平台或使用BackTrack、Kali这类完善的环境对自己感兴趣的一个又一个无线热点进行安全评估，体会那种突破无线密码、接入目标无线网络的成就感。

随着无线攻防手段的更新，攻击方法也有了更有趣的发展，如先侦测无线客户端发出的曾经连接过的热点的Probe信息，再通过软AP程序产生相同名字热点的所谓EvilAP的钓鱼攻击方法，将无线目标拉入虚假的无线环境进而发起攻击，或者窃取目标网络流量中的敏感信息。独角兽团队所支持的2015年315晚会上WiFi安全环节的出现，也让圈里的老无线安全研究者的心为之一动。这个安全演示环节出自

DEFCON黑客大会Wireless Village上有名的The Wall of Sheep（绵羊墙），这个项目是为了教育人们—“你很可能随时都在被监视”，同时也给那些参会的人难堪，参加安全大会还如此不注意安全，难怪被贴到绵羊墙上。绵羊墙的账号和部分隐匿的密码将被投影在专门的一个会议室内的大屏幕上，The Wall of Sheep大约由7名来自北美的安全人士维护，他们每年花2周时间来拉斯维加斯参加DEFCON。大会提供免费的“hostile”的网络（BlackHat和DEFCON提供的无线网络）供参会者接入访问，如果接入了，那么你的所有网络活动都可能被监听和探测。

1.1.2　无线安全与移动安全的区别

现在安全行业对“无线”及“无线安全”的概念是略模糊的，多数从业者认为无线即指手机无线端，无线安全则指手机系统安全、App端安全。这其实是不正确的，这个安全领域更应该定义为移动安全。本书所提到的无线安全则是更广义的，是指所有使用无线通信协议的无线电技术的安全，即“无线电安全”。

1.1.3 无线安全的现状

随着手机的普及和人们日益增长的随时随地无线上网的需求，针对WiFi的各类攻击屡见不鲜，从国内各类媒体对用户在咖啡厅等公共场所上网被钓鱼事件的报道就可以发现，WiFi安全已经是一个社会安全问题，手机厂商、安全公司也都对WiFi造成用户隐私信息泄露等问题在各自能把控发力的地方下足了工夫。如苹果公司在iOS8系统上增加了新的安全特性，可以使设备的无线MAC地址随机化，用来躲避在使用WiFi的过程中暴露自己手机的“物理指纹”；安全厂商也在各自的手机卫士类产品内增加了对周围无线热点评估的功能，以谋求提高用户连接WiFi时的安全性，使用户不受黑客无线钓鱼攻击的威胁。

那么我们只关注WiFi安全就可以了吗？答案是否定的，随着物联网（IoT）的持续蓬勃发展，现在的手机、智能设备对各类无线模块、传感器的需求越来越大，蓝牙、GPS、NFC模块早已成为必备项。此时此刻，我们从安全的角度来看，整个行业对于使用这些无线技术的安全准备是不足的。

美国Todd Humphreys教授领导的无线导航实验室，是GPS安全研究领域非常领先的一个团队。早在2012年，Todd Humphreys教授就在TED发表了演讲，呼吁公众注意GPS安全。在DEFCON23上，中国独角兽团队的安全研究员黄琳向全球展示了如何使用低成本的软件无线电设备欺骗手机、汽车甚至无人机。Todd Humphreys教授在2016年2月又以文章Lost in Space：How secure is the Future of Mobile Positioning？再次提出了他对于未来依赖GPS技术的设备在受到GPS欺骗攻击时抗攻击能力的疑问，以及这种攻击手段被滥用的担忧。

2016年2月18日，是苹果公司寄予厚望的Apple Pay入华首日，但其实在几天之前，国内安全行业的专家们就已收到各类媒体关于Apple Pay这种NFC的交易方式是否安全的咨询。

从以上两个事件可以发现，在传统的无线局域网安全之外，更多的“无线”安全进入人们的视野和生活，我们已无法视而不见。安全从业者必须要具备在该领域安全的架构设计及风险评估能力。

1.2　无线安全攻防思路

1.2.1　常见攻击对象

一张门禁卡、一把无线钥匙、一个无线遥控器、一部手机、一辆汽车、一台无线呼吸监测仪、一架飞机，甚至一辆坦克、一颗卫星，只要攻击对象使用了无线介质进行数据交互，那么这条无线链路就有可能被监听、解密、重放、欺骗、劫持，甚至被入侵、被控制。看似不可能直接接触的目标，往往在无线通信层，攻击会变得如此直接。

作为一个安全评估人员，首先要做的就是确定可以评估的攻击面。以一辆汽车举例，汽车的无线钥匙系统和汽车的胎压传感系统（TPMS）多数采用315MHz或433MHz的RF传输数据；如果是无钥匙启动系统的汽车，则多会采用125KHz或13.56MHz的RFID技术；这辆汽车如果还具备Telematics网络连接能力，那么车机控制系统一定会具备2G到4G的蜂窝通信能力，并且本身一定还会提供2.4amp;5.8GHz的汽车WiFi网络，这些都是暴露在外可进行风险评估的攻击面。

1.2.2　无线安全攻击手段

无线攻击不同于传统的Web攻击等手段，它是一种靠尝试介入无线通道为起点，最终获取连入该无线通道并实施信号控制，或者利用网络进行更深入的渗透测试的攻击形态。对于这样的无线通信安全评估，大致可分为如下几种手段。

攻击手段之一：无线数据报文监听。使用与目标无线系统运行频率相同的监听设备对全量无线报文进行收集及数据逆向分析、解密。如监听WiFi使用无线网卡，监听蓝牙使用蓝牙嗅探设备，监听无线钥匙则使用SDR设备。将无线报文数据通过相应的方法解密后，可以深入了解整套无线系统的运作原理，找出关键的无线指令。

攻击手段之二：无线信号重放攻击。如果目标系统的无线通信协议没有设立有效的时间戳或随机性等防信号重放机制，那么当使用相应的无线设备截获一段合法合规的无线指令时，就可以通过将这段信号指令直接重放出来，影响目标系统。如截获了无线钥匙的开门指令，可以不使用钥匙，直接重放开门或关门信号，就能获得打开目标车门或其他设备的权利。

攻击手段之三：无线信号欺骗攻击。联合无线监听及解密，通过掌握目标无线协议的报文构成及关键密匙、校验方法等，直接构造合法的可通过协议认证的无线报文，影响目标无线系统的运作。

攻击手段之四：无线信号劫持攻击。通过使用协议层（如压制WiFi热点的软件MDK3）或通信层（如发出特定频段噪声的信号干扰器）网络阻断的方法，也称无线拒绝服务攻击，将目标网络环境从合法领域拉入可控的虚假领域，再通过劫持上下行无线网络流量进行各类攻击。如使用MDK3压制一个无线客户端到一个合法无线热点的连接，迫使其接入虚假的无线热点；又如使用3G到4G信号干扰器，将手机或汽车的蜂窝网络从相对安全的网络压制到不够安全的2G网络环境，再被开源基站控制，继而进行上下行流量的中间人劫持攻击。

1.2.3　无线安全防范思路

无线安全防范要从攻击面去考量，可靠的通信协议、强认证、强通信加密、抗信号干扰是一套无线通信协议安全与否的核心。具体问题要具体分析，关于各类无线通信协议详细的防范方法请读者阅读后续章节。

1.2.4 无线安全趋势

如今，随着RTL-SDR、Ettus的USRP及HackRF、Nuand的bladeRF等各类软件无线电设备的价格下降、软件环境社区的完善，安全研究者已经可以轻易地拥有一块无线频谱覆盖50MHz～6GHz的无线信号分析“神器”，无线黑客们已经不再像当年只能利用2.4GHz的无线网卡进行狭窄的“无线”攻防。不得不说的是，廉价的SDR设备推动了无线安全领域的发展，安全研究人员借助这些设备已经从传统安全领域跨界到无线电安全领域，但与此同时也增加了被恶意利用的风险。

刀可以用来杀人，也可以用来救人。这种设备的民用化，或许导致被更多的怀有恶意的黑客所利用，以致影响、威胁到我们的生活，但同时我们也需要使用这些设备来了解、评估已经在使用的无线技术。攻与防一直在博弈，我们只有与时俱进，了解最新的攻击手段，才能优化改良现有的体系，使我们的生产生活更加安全。当年科学家们认为6位数字的密码已经足够安全，但随着计算机CPU运算速度的提高，现如今密码的复杂度如何想必大家也有目共睹。就像GPS欺骗攻击这种无线信号攻击方式一样，当年研发GPS的科学家们怎么也不会想到，时至今日，一个普通人也可以利用一些廉价的设备发出这些科学家当年自认为只有他们的卫星才能发出的导航信号吧。

无线电技术下的攻防在未来会愈演愈烈，各种资料、设备、研究社区的丰富也使得更多的安全研究者开始踏入这个领域进行各自的研究，无线电通信安全终将成为信息安全体系重要的一环。

第2章　RFID智能卡的安全研究

在过去的一段时间内，越来越多的企业采用RFID和非接触式智能卡来替代手输密码、磁条卡或者纸制票卡。非接触式智能卡包含了一小块可以通过无线通信方式来存取的内存，但是跟RFID标签不同的是，非接触式智能卡同时具有一定的运算能力。大部分非接触式智能卡都会采用类似于对称加密的加密方法，来限制相对应的程序对非接触式智能卡的内存控制权限。

很多大范围系统都会使用非接触式智能卡。这种卡一般会应用在公共交通运输系统中，例如伦敦公共交通系统的Oyster卡、北京公交地铁卡，采用的都是非接触式智能卡。很多国家已经将非接触式智能卡芯片纳入电子护照标准。一些办公楼甚至安全重地（飞机场、军事基地）现在都采用非接触式智能卡作为准入权限控制手段。

市场上有很多种非接触式智能卡，这些卡的大小、外壳、内存大小、计算能力都不同，此外，在卡本身提供的安全特性上也有很大的不同。一种应用非常广泛、知名的卡就是Mifare。Mifare是NXP半导体公司（前身为飞利浦半导体）生产的一系列非接触式智能卡的统称，目前这个系列包含有如下几种类型的智能卡：Ultralight，、Classic、DESFire和SmartMX，它们的用途非常广泛。NXP半导体公司的Mifare系列大约占据了全世界80%的非接触式智能卡市场。Mifare Classic类型的智能卡均遵循ISO/IEC14443Type A标准，通过一种叫作Crypto-1的流加密方式提供了相互校验以及数据加密的安全特性。这种加密方式是NXP半导体公司持有的专利，并且设计细节均处于保密状态。

2.1　Mifare Classic智能卡简介

Mifare Classic智能卡一般有两种，即Mifare Classic1K和4K。区别在于，1K卡有1KB大小的EEPROM内存，分成4个扇区，每个扇区包含16个区块，一个区块大小为16字节（Byte）；而4K卡提供了4KB大小的EEPROM内存，分为4个扇区，每个扇区包含32个区块，还有余下16个扇区，每个扇区包含8个区块，共计256个区块，每个区块的大小同样为16字节。4K卡的扇区排列情况如图2-1所示。

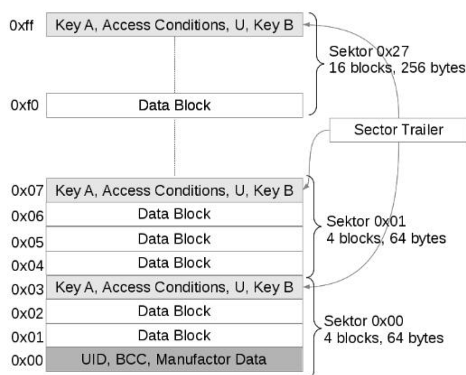


图2-1　4K Mifare Classic卡的扇区排列情况

Mifare Classic智能卡的第一个区块包含了一个独一无二的标识序列号（UID），这些数据根据供应商不同而不同，并且这个区块在正规出厂后是会被写保护的，无法进行二次擦写。每个扇区的最后一个区块，存储着访问Key和访问控制条件，这个区块并不存储普通用户数据。读卡器在对卡的某一扇区进行任何内存操作之前，都要经过这个扇区的身份验证流程。因此，设计人员在扇区的最后增加了一个尾区块，这个区块包含了验证流程中要使用的访问Key（Key A和Key B）。访问控制条件同样存储在这个区块中，限定了读卡器对这个扇区能执行什么类型的内存操作。

扇区的尾区块包含了两个Key，即Key A和Key B。Key A永远无法被外部读取，但是Key B可以被配置为可被外部读取以及不可被外部读取。当Key B被配置为可被读取时，只有Key A会被用于身份验证，Key B将只用于存储数据。同时，还有一个空余的字节U，不存储

有效数据，这个字节的位置如图2-1所示。

2.2　Mifare Classic智能卡安全分析

我们前面提到Mifare Classic智能卡采用的加密方式是Crypto-1加密算法，此加密算法是基于HITAG RFID系统中使用的HITAG2算法的。来自柏林混沌计算实验室（Chaos Computer Club，CCC）的Nohl和Plotz两位安全研究人员对Mifare Classic智能卡进行了硬件上的逆向分析工作，并且暴露了一系列的安全问题。

在CCC实验室对Mifare Classic智能卡进行逆向分析工程之前，业界对于这种智能卡最有效也最常见的攻击方式就是简单的暴力穷举攻击。由于卡内部使用的加密算法属于未知状况，所以这种攻击方式是对卡本身的工作模式进行的攻击。由于智能卡不具有电源，所有的工作能源都来自于读卡器远程提供，而向EEPROM中写入数据又会消耗太多的时间，所以卡并不会在某个地方记录对其进行的错误口令尝试次数。

这种暴力破解的攻击方式最大的问题就是时间。一次Key尝试大概需要5ms，要对48位的Key进行全部尝试的话，则需要55000年才能找到正确的那个Key。如果把整个过程搬到计算机上来进行穷举攻击还是有可能的，但是由于卡加密和验证算法是保密的，所以想要对智能卡发展出一种有效的攻击方式，必须要获得加密算法。

在此之前，业界也有过对算法进行逆向还原的案例，比如GSM通信协议中的A1/A2算法、车辆远程控制中用的HITAG2和KeeLoq算法。但是这些算法都是在软件中使用的，逆向分析比较方便。而对于Mifare芯片，所有的计算都在硬件中完成，没有软件协助。

Mifare芯片对于安全研究人员来说就是一个黑盒，在黑盒安全分析中，一般采用输入和输出对照的方式来进行黑盒算法推算，但是Mifare采用的加密密钥有48位，很难猜出应用了什么算法。

最后摆在CCC实验室面前的难题就是，Mifare芯片并不对错误的Key进行回应。按照研究人员Henryk Plotz的说法，这也是Mifare Classic智能卡为数不多的合格设计点之一。

结合以上各种攻击手段的可能性和优缺点，研究人员决定从硬件逆向入手。

2.2.1　RFID芯片硬件逆向分析

RFID芯片与RFID芯片卡是不同的，RFID芯片卡包含了一个很小的RFID芯片，这个芯片被一圈天线和塑料包裹，最终形成一张信用卡大小的塑料卡片。因此，研究人员使用丙酮对自己的RFID芯片卡进行处理，从而溶解出RFID芯片。RFID芯片大小约为1mm×1mm，对于Mifare Classic卡来说，这样一个小小的芯片包含了7层电路板。为了分析这些电路板层，研究人员购买了一个价值€1000的配有摄像机的显微镜，这个显微镜可以将拍摄到的画面直接传送到计算机中，方便进行下一步分析。

虽然使用显微镜可以详细观察电路状况，但是问题在于只能看到芯片的第一层电路板；其余层的电路板虽然可以透过第一层电路板隐隐约约看到，但是肯定不能用来推测电路实际状况。所以，需要一种手段来剖出全部的电路板层。业界普通做法是采用化学蚀刻方式。但是为了尽可能降低研究的复杂度，CCC实验室直接采用了机械打磨方式。为了打磨电路板，研究人员采用了抛光乳液和0.04μm的砂纸进行打磨工作。

在打磨过程中，为了能看到每一层电路的详情，所以打磨一定要很均匀。但是由于芯片有天线电路，以及其他一些连接点，因此芯片本身就凹凸不平。此外，芯片大小也就1mm×1mm，每一层电路板的厚度只有1μm，徒手操作十分困难。CCC实验室把芯片周围重新熔铸到塑料里，这样可以方便打磨。CCC实验室花费了大约两周的时间，一共磨坏了10块RFID芯片，最后终于将7层电路板全部打磨出来并用显微镜详细拍照。

在详细拍摄了RFID芯片的内部构造之后，接下来就需要对芯片的电路结构进行详细分析了。在这块芯片上，大约有10000个、70余种门电路需要分析。为了加速逆向工作，研究人员特地开发了一个叫作“degate”的Matlab脚本，用于对显微镜拍摄到的电路图像进行门电路详情分析。脚本处理完成之后，会在照片上标识出哪里用了哪种类型的门电路。图2-2即展示出了未经脚本处理的原始照片和经过“degate”脚本处理之后的照片对比。

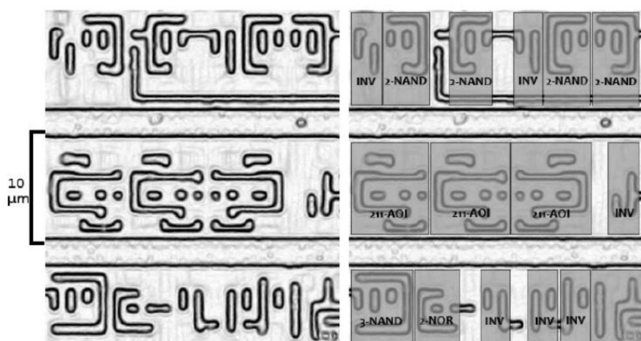


图2-2　“degate”脚本处理前后的电路照片对比

有了经过标识门电路的显微照片之后，研究人员开始对门电路之间所有的连接线路进行人工分析。每一层电路大约包含100行和100列的连接点，因为数量太多了，所以很难对整块芯片进行系统性的完整分析。因此CCC实验室就把研究重点放在了加密逻辑电路上，加密模块大约只占整体电路的10%，大大减少了工作量。先前对显微照片的处理很有利于定位加密逻辑电路在芯片上所处的位置，因为一般异或电路（XOR）和电平反转电路（flip-flop）都会在加密逻辑电路中大量出现。如果芯片的某个位置出现了大量的以上两种门电路，那么就意味着这个区域用于加密计算的可能性很大。根据Mifare Classic智能卡使用的48位长的密钥，如果发现48个连接在一起的异或门电路，那么就意味着这附近的一片集成电路就是芯片的加密模块。

加密模块大约由1000个门电路和1500个连接点构成。为了彻底绘制出加密模块的电路图，研究人员逐个跟踪连接点的连接状况，手工绘制出加密模块的原理图。这个过程非常消耗时间，比如有的连接点会横跨三个电路层，甚至整块芯片。

最终经过两年的时间，研究人员彻底逆向绘制出了Mifare Classic智能卡的加密算法[注]，具体细节将在下一节中阐述。

1. http://proxmark.nl/files/Documents/13.56%20MHz%20-%20MIFARE%20Classic/The_MIFARE_Hack.pdf返回

2.2.2　RFID芯片加密算法细节

在介绍Mifare Classic智能卡的加密算法之前，我们应明白加密算法要有输入和输出。所以，一个首要的问题就是，Mifare Classic加密算法的输入是什么？为了研究出加密算法的输入和初始条件，CCC实验室的研究人员采用OpenPCD作为读卡器进行了测试。之所以采用OpenPCD，是因为这是一个开源的读卡器，研究人员可以详细观测到所有收发的原始数据。

就像我们前面所提到的一样，卡不会对错误的验证密钥进行响应。因此，为了进一步明确加密算法的初始条件，研究人员向卡发送了正确的验证密钥。之后，对密钥逐位进行改变，来观测智能卡是否依然给予响应。经过不断的尝试，研究人员发现智能卡的UID和验证密钥同时用于加密算法的输入。当然，对于CCC实验室来说，这并不是什么重大发现，因为此情况已经写在了Mifare Classic智能卡文档中。通过对智能卡的硬件逆向分析，研究人员得知加密算法逻辑电路的输入源只有一个，但是UID又和验证密钥一同作为输入，因此UID必然和验证密钥以某种形式连接在一起。研究人员猜测，UID和验证密钥以异或方式组合在一起，然后作为加密逻辑电路的输入源。为了验证这一猜测，他们把UID和验证密钥的第一位都进行了位反转，这样如果加密逻辑电路的输入源是UID和验证密钥的异或组合，那么就会得到同样正确的验证结果。研究人员将想法付诸行动之后，发现位反转后的验证同样给出了验证通过的响应，这说明上面的猜测是正确的。

在对Mifare Classic智能卡无线通信协议分析的过程中，CCC实验室的研究人员发现算法使用了一个线性反馈移位寄存器（Linear Feedback Shift Register，LFSR）来实现伪随机数发生器（PRNG）。

线性反馈移位寄存器（LFSR）是指给定前一状态的输出，将该输出的线性函数再用作输入的移位寄存器。赋给寄存器的初始值叫作“种子”，因为线性反馈移位寄存器的运算是确定的，所以由寄存器所生成的数据流完全决定于寄存器当时或者之前的状态。而且，由于寄存器的状态是有限的，所以它最终肯定会是一个重复的循环。

在一个被动式的RFID芯片中使用LFSR的问题就在于，芯片由外部读卡器提供电源，这就意味着，每次芯片从读卡器附近移开又重新

上电时，用于伪随机数发生器用途的LFSR都会重新初始化，并且产生与上次一样的随机数。CCC实验室为了证明这个理论做了一个实验，将智能卡放置在读卡器上，然后重启读卡器。最后发现，从读卡器上电到智能卡产生随机数的时间间隔几乎相同，并且在27次尝试中，有12次智能卡产生了相同的随机数。

研究人员Gans等声称，他们可以做到每小时产生60万个随机数，其中每个随机数至少重复了4次，也就是说，每个随机数都会在0.618s后重复一次。所以，对Mifare Classic智能卡的一种有效攻击方式就是重放攻击。对于如何进行重放攻击的具体解释，我们会在后续的章节中描述。

CCC实验室绘制的加密算法逻辑电路的原理图如图2-3所示，Crypto-1算法包含了一个线性反馈移位寄存器（LFSR）和一个过滤函数f(x)。

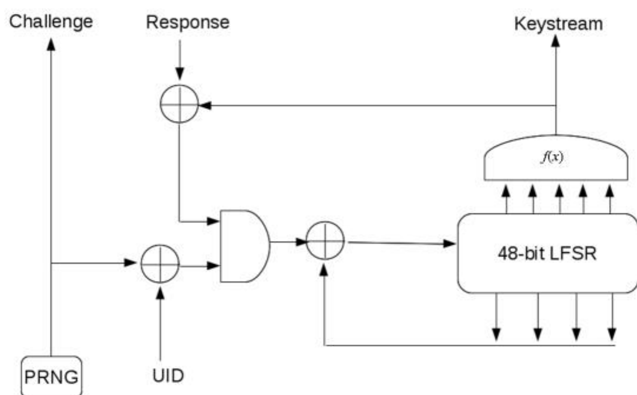


图2-3　Mifare Classic加密算法逻辑电路原理图

在算法的输入部分，移位寄存器使用验证密钥进行初始化。最后，伪随机数发生器产生32位二进制数字，并与智能卡的UID进行异或操作，得到的结果会逐位填充到接下来的移位寄存器中，得到的随机数结果同样用于读卡器和智能卡之间的挑战-响应机制中。在这里，加密算法开始发挥作用，所有的输入和输出都将被密钥流进行异或加密。移位寄存器本身只用于存储数据，或者用于计算循环冗余校验（CRC）。

在算法初始化过程中，48位的加密密钥加载到移位寄存器中，然后由UID和PRNG异或出字串流（UID即为智能卡的标识符，PRNG表示伪随机数发生器产生的随机数）及PRNG产生的随机数同时发送到

读卡器，作为挑战-响应机制中的第一个挑战，读卡器随后发送响应证明读卡器有正确的读取Key。在一些针对读卡器的攻击中，攻击者只需要与读卡器进行交互，所以其伪装成智能卡，用于挑战的“随机数”实际上是刻意选中的，而非真实产生的随机数。

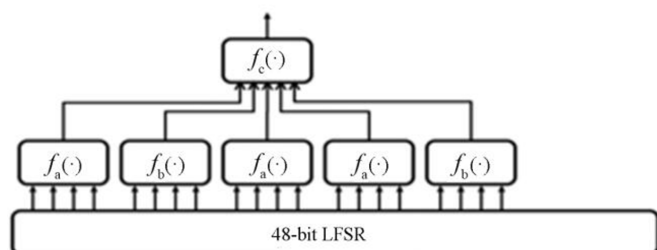


图2-4　过滤函数f(x)的构成

在每个时钟周期内，过滤函数f(x)都会使用LSFR中的20位计算出密钥流的1位。函数f(x)实际上由3个小规模函数的6个实例组成（见图2-4），这些函数实际上具有统计学偏移性（Statistically Biased）——如果一直保持着同一个输入，那么输出结果（1或0）的概率并不是均等的（50%）。笔者没有找到关于过滤函数更详细的信息，感兴趣的读者可以进一步搜寻、研究。

2.2.3 Mifare Classic业界破解过程回顾

2007年年底在柏林举行的第24届黑客大会Chaos Communications Congress上，有两位研究者宣读了他们合作的论文。其中一位是德国的学者Henryk Plotz，另一位是弗吉尼亚大学的在读博士Karsten Nohl，他们利用显微镜，对Mifare Classic芯片进行了抽丝剥茧般的逐层分析，然后辅以RFID读卡器，得到了Mifare Classic芯片每层的布线图，从而分析出芯片中近万个逻辑单元，分别是逻辑电路的与门、或门以及触发器。

他们从这些逻辑门中发现了规律，从而大大简化了分析的进程。接下来就是要找到其中关键的加密处理部分，他们按照自己获得的数据信息进行了重构，在此过程中他们详细分析了各个单元模块的功能。经过一番辛苦劳作之后，他们发现了Mifare Classic芯片的若干安全隐患。其中之一就是他们掌握了一个16位随机数发生器的原理，每次都能够正确地预测出下一个随机数的值。

2008年2月，荷兰政府公布的一项报告对此事进行了回应，报告肯定了Karsten Nohl和Henryk Plotz的发现，但却断言由于攻击成本太大，Mifare卡系统在两年内还是安全的。他们估计攻击所使用的硬件成本约为9000美元，并费时数小时。

Karsten Nohl马上又发表了一篇名为Cryptanalysis of Crypto-1的文章，公开了Mifare Classic加密算法Crypto-1的核心，就是一个48位的线性反馈移位寄存器加上几个输出函数过滤器。而这几个函数过滤器应用了3种不同的函数关系，它们都有统计学偏移性的弱点，利用这些弱点和之前掌握的随机数发生器的知识，使用普通计算机通过向读卡器发送几十个挑战数，就能够猜出卡密钥中的32位是什么。其中有12个比特仅通过乱码流的第一位比特值便可判定，其余36位密钥值的破解时间，若使用一个FPGA装置是在30s内，即使使用普通PC也就用几分钟。

不过，Karsten Nohl所公布的信息还是有相当保留的，他虽然介绍了攻击方法，但是却没有技术细节；对Crypto-1的内容和缺陷也披露得颇为朦胧。例如，并没有指出48位的线性反馈移位寄存器的20个抽头的具体位置，也没有透露3个输出函数的细节，更没有任何他所使用的硬件辅助工具的信息，所以我们也不知道除了PC外，攻击的硬

件成本到底是多少。至此，Mifare Classic的最后一层面纱还没有在公众面前揭开。

于是，荷兰Radboud大学计算与信息科学学院的研究人员开始出场。先是这个学院的Gerhard de Koning Gans等人组成的一个小组公布了他们的论文《对Mifare Classic的一种实用的攻击方法》，这篇文章研究了Mifare Classic卡的内部结构和卡与读卡器之间的通信协议，提出了一个实用的、低成本的攻击方法，能够从卡的存储器中获取私密信息。“由于伪随机数发生器存在一个缺陷，我们能够还原由Crypto-1序列密码算法产生的乱码流。我们利用流密码的延展性去读卡上第一个扇区中所有的存储块。并且，如果知道卡上任意扇区的一个存储块，我们就能读这个扇区。最后，也许还可制造更多的威胁，例如修改存储块……”

从密码学角度上说，这个小组的工作还很不彻底，他们没有破解任何一个卡的密钥；但又确实很实用，因为使用了很简单的原理和方法就读出了卡上的许多私密信息。“在这种攻击中，我们不需要知道Crypto-1算法，只需要知道它是一个按位加密的流密码即可。”另外，他们还介绍了自己所使用并加以改进的廉价的射频截听工具ProxmarkIII，它能够模仿卡或读卡器与对方的通信，价格约几百美元。拥有了它，不但可以获取并制造大量的报文，而且使攻击简便易行，且成本变得十分低廉。

国内有人把这种方法称为“重放攻击”，因为是截听了读卡器与卡之间的射频信号，然后再向卡重放修改过的信息而达成的。

不过，大多数黑客或别有用心的人还是会失望，因为到这个阶段他们还不知道怎么去克隆一张Mifare卡。

荷兰Radboud大学的这个小组并没有停止他们的工作，反而增加了人员，加快了速度。他们的下一篇文章《拆解Mifare Classic》终于向公众揭开了Crypto-1的庐山真面目。

该文章声称：在逆向研究了Mifare Classic卡的安全机制，包括认证协议、对称加密算法和初始化机制后，他们发现了存在于上述机制中的几个安全漏洞，利用这些漏洞，可实施两种攻击，这两种攻击都可以从一个真实的读卡器中获取密钥。其中一个攻击仅需要与读卡器进行一次或两次的认证尝试，不到1s。“采取相同的方法，攻击者可以侦听到卡与读卡器之间的通信内容并将其解密，而尽管可能涉及多重认证。这使得攻击者可以克隆一张卡，或者将卡的内容恢复到之前的状态。”

这次他们使用的工具除了ProxmarkIII外，又尝试了更为廉价但功能相当的一款叫作GHOST的自制仪器，这个设备仅估价40欧元，比ProxmarkIII又低了一个数量级。

与Karsten Nohl和Henryk Plotz的做法不同，荷兰的科学家们公布了所有细节，包括Mifare Classic芯片的算法逻辑和他们自己设计的攻击手段。

至此，荷兰Radboud大学的研究者向世人公布了3种攻击方法。然而，他们并没有停下脚步，之后又发表了文章Wirelessly Pickpocketing aMifare Classic Card（可以译为“无线窃取M1卡”）。

在《无线窃取M1卡》一文中，研究者阐明了M1卡在报文产生奇偶位和所谓嵌套认证两个方面的漏洞；利用此漏洞，攻击者可以通过工具仅仅研究该工具与一张M1卡之间的通信数据便可以成功破解该卡的所有密钥，从而克隆这张卡。

请注意，与以往公开的任何一种针对M1卡的攻击所需要的条件不同的是，这次的4种攻击方案都不再需要一个“真实的”读卡器。而他们此前介绍的3种方法都需要一个真实的应用系统中的读卡器，在攻击时要截取它与被攻击卡之间的通信报文作为分析的资料。也就是说，使用以前的方法，如果想攻击北京的公交卡，你就得抱着一个仪器去公交车上或到地铁站的收费口前晃悠。这难免不被人看到，做贼又心虚者没准就会感到太冒险而放弃了。从这个角度来讲，这些攻击对应用系统的威胁就不那么大了。而新的攻击方法就不同了，你可以买张公交卡回家自己琢磨去，没人知道你要干什么。对攻击者越安全，对被攻击物就越危险！所以，这样的攻击更加实用、更具危害性。

2.3 Mifare Classic智能卡破解实例

与常见的Mifare Classic智能卡破解、复制的例子不同，这里的实例将采用ProxmarkIII和Chameleon-Mini配合的方式，实现对智能卡的破解以及多卡复制。

2.3.1　ProxmarkIII简介

ProxmarkIII是一款最初由Jonathan Westhues开发的开源硬件产品，用于窃听（双向）、读、写、模拟、克隆RFID芯片卡。通俗地说，这款产品就是一个用于测试高/低频RFID系统安全性的器件。

ProxmarkIII几乎可以做低频（~125kHz、134kHz）或者高频（~13.56MHz）下的任何事情，可以作为读卡器、写卡器，可以监听一个合法读卡器和正常RFID智能卡之间的通信内容。因此，用户可以通过ProxmarkIII保存监听到的通信信号，供后续进行仔细分析，或者直接伪造出自己的卡信号，从而模拟出一张RFID智能卡。

如图2-5所示是ProxmarkIII裸板近照。

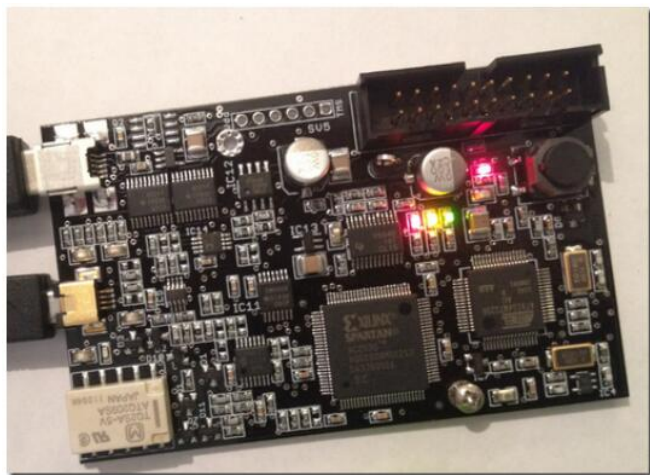


图2-5　ProxmarkIII裸板近照

其中，最大的那块芯片（IC）是一片FPGA，位于它右侧的是一片ARM7，ARM7用于执行在Flash中存储的代码，并且可以通过USB来刷写内部的固件代码。USB连接头位于照片的右上角，在它下面的连接头是用于连接天线的连接端口。

如果想要对高频、低频的场景都进行攻击测试，那么就需要两个天线，一个在高频下使用，一个在低频下使用。

如图2-6所示是一块高频天线板，通过一条USB线与ProxmarkIII主板连接在一起。

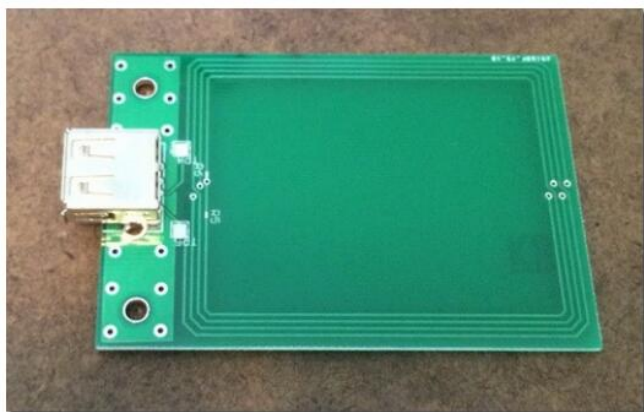


图2-6　与ProxmarkIII配套的高频读卡天线

如图2-7所示是一块低频天线板，其使用方式与高频读卡天线相同。



图2-7　与ProxmarkIII配套的低频读卡天线

ProxmarkIII是一个开源的安全设备，故其内置的固件也因开源而不断地进行升级及修改，如果读者打算在自己的ProxmarkIII上使用最新的固件，则可以到https：//github.com/Proxmark/proxmark3下载最新的固件代码。同时你会发现这是一个非常活跃的技术交流社区，几乎每周都会有新代码和新特性可以用于更新。

为了使自己的固件拥有最新的特性，建议读者对ProxmarkIII进行客户端和硬件固件的定期更新工作。下面我们以将ProxmarkIII从libusb版本升级到USB CDC版本为例进行固件烧写演示。

2.3.2 ProxmarkIII固件烧写及使用

在烧写之前，有一点需要提及，因为ProxmarkIII的官方固件在r630版本之前，一直采用的USB驱动为libusb模式，在r630及以后的版本中，官方采用了USB CDC模式。在r629之后的版本中，如果要使用客户端中自带的烧写工具对固件进行烧写，那么就只能采用USB CDC模式，因此对于r629及以前的版本，要及时进行bootrom的更新。

下面介绍通过JTAG接口使用J-Link对ProxmarkIII进行bootrom及固件的烧写。

通过J-Link烧写固件需要如下几个设备：

- J-Link V8及配套线缆
- ProxmarkIII
- USB Mini-B线

1.固件编译

值得一提的是，ProxmarkIII的官方Wiki㊟ 非常详细，如果在使用过程中有什么问题，到Wiki进行搜索多半可以找到自己想要的答案。

在固件编译的过程中，我们首先要到ProxmarkIII的GitHub官方地址㊟ 将最近的源代码克隆到本地。然后在ProxmarkIII的Google Project下载地址 ㊟ 下载固件编译依赖包，这里依赖压缩包名为ProxSpace-130613.7z，并且依赖包的提示为ProxSpace for the new USB CDC interface，如图2-8所示。



图2-8　ProxmarkIII编译依赖包ProxSpace提示

解压缩ProxSpace-130613.7z到C：\Proxmark\ProxSpace，将从

GitHub上克隆下来的最新源代码文件夹重命名为pm3，复制并且覆盖ProxSpace文件夹下的子文件夹pm3，编辑C：\Proxmark\ProxSpace\runme.bat，更改“set MYPATH＝%~dp0”为“set MYPATH＝C：\Proxmark\ProxSpace\”，最后一行可能是“msys/msys.bat”，需要改成“msys\msys.bat”。双击runme.bat，这时如果前面的步骤都没有问题的话，就会正常出现GNU终端窗口，如图2-9所示。

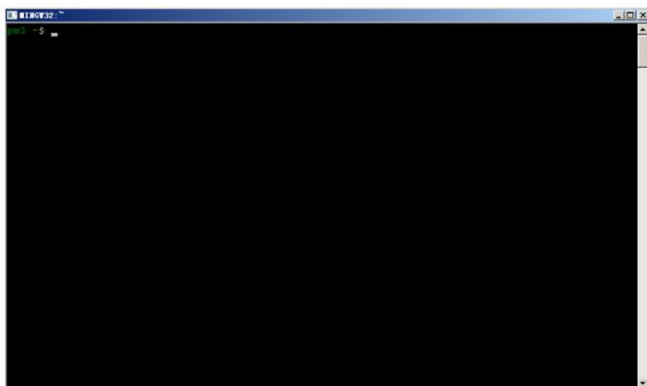


图2-9　GNU终端窗口

此时，如果窗口未出现报错信息，则执行“make cleanamp;amp;make all”即可进行固件编译工作。

2.固件烧写

在固件烧写时，下载并安装好J-Link ARM，将PC、J-Link、ProxmarkIII通过烧写线缆连接在一起，打开J-Flash ARM软件，然后选择“Options”→“Project settings”，如图2-10所示。

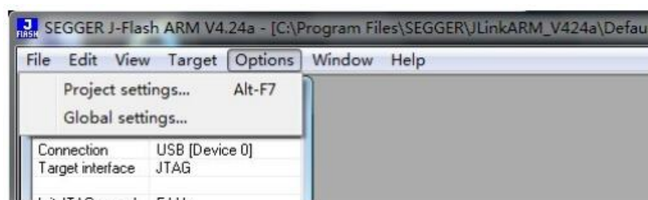


图2-10　J-Link ARM界面

选择好ProxmarkIII采用的主控芯片，切换

到“CPU”下，“Device”选择“Atmel AT91SAM7S256”，“Clock speed”选择“Auto detection”，其他采用默认，最后确定保存，如图2-11所示。



图2-11　工程参数设置

如图2-12所示，选择“Target”→“Connect”连接设备，连接成功时ProxmarkIII板上的4个指示灯会同时亮起，如图2-13所示。

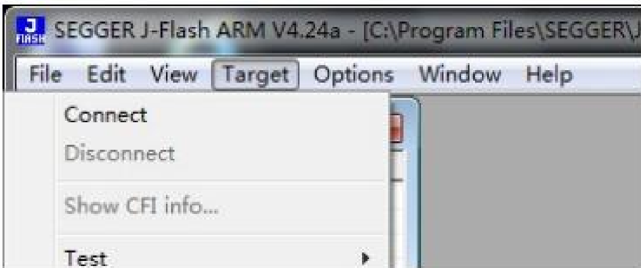


图2-12　连接ProxmarkIII

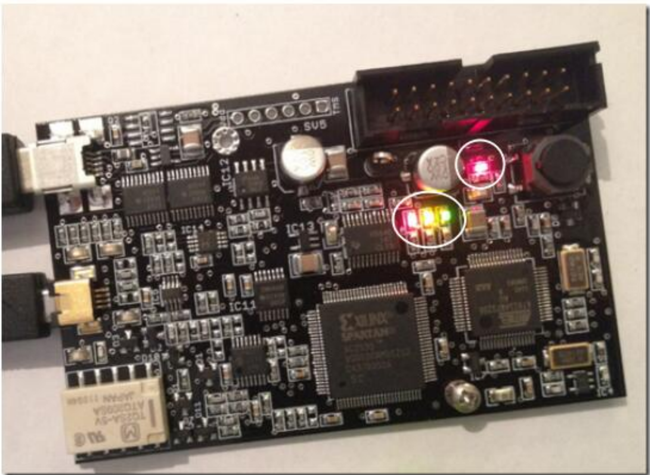


图2-13　ProxmarkIII板上的4个指示灯全部亮起

连接完成后，可以选择“Target”→“Erase chip”将原始数据擦除干净。

选择烧写的文件，选择“File”→“Open data file”，打开C：\Proxmark\ProxSpace\pm3\bootrom\obj\bootrom.s19，如图2-14所示。

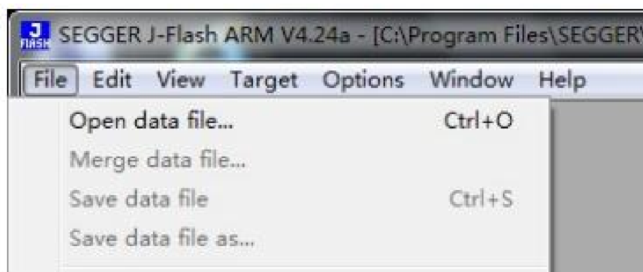


图2-14　选择编译好的bootrom

开始烧写，选择“Target”→“Auto”，烧写成功断电，去掉J-Link重新通电，会识别出新硬件，这时记录下ProxmarkIII对应的端口号，如图2-15所示，即为COM4。



图2-15　查找ProxmarkIII对应的端口号

打开CMD窗口，执行命令“cd C：\Proxmark\ProxSpace\pm3\client”。

断开ProxmarkIII，按住板子上的按键，连接ProxmarkIII，等到黄色、红色LED同时亮起时松开按键，在CMD窗口中输入“flasher COM4-b..\bootrom\obj\bootrom.elf ”，成功之后断开连接。这里要注意的是，命令中COM后的数字，即是上两步中实际ProxmarkIII对应的端口号。

按住板子上的按键，连接ProxmarkIII，等到黄色、红色LED同时亮起时松开按键，在CMD窗口中输入“flasher COM4-b..\armsrc\obj\osimage.elf”，成功之后断开连接。

以上操作顺利完成之后，ProxmarkIII就已经升级到USB CDC版本的最新固件了。

1. https：//github.com/Proxmark/proxmark3/wiki/Windows返回

2. https：//github.com/Proxmark/proxmark3返回

3. http://code.google.com/p/proxmark3/downloads/list返回

2.3.3 ProxmarkIII客户端

ProxmarkIII在正常工作时，上位机客户端将用户命令传输至ProxmarkIII硬件，硬件在运算之后，将运算结果返回至客户端并显示给用户。由此可见，ProxmarkIII配套的客户端程序并不进行任何运算行为，只负责传输数据并进行展示。所以在ProxmarkIII使用过程中要注意的一个重要的地方就是，在中止操作时，在客户端按下“Ctrl＋C”键或者强制退出没有意义，只能通过按下硬件板上的按钮取消操作。

此外，ProxmarkIII的客户端和固件是配套的，所以在正常使用时，最好使用与当前固件版本相对应的客户端程序。如果是手动自行编译刷写的固件，那么编译时将自行生成配套的客户端程序，直接使用即可。

对于USB CDC版本的固件，双击Proxmark工程文件夹中的runme.bat，在出现的GNU终端窗口中，输入“./client/proxmark3.exe comX”，命令中的X即为ProxmarkIII接口在系统中对应的端口号，命令成功执行之后，将会看到如下命令提示符：

```
proxmark3>
```

一旦连接成功，输入“hw version”即可检查当前固件的版本。如果固件版本低于r798，那么应该会看到类似于如下的输出：

```
db# Prox/RFID mark3 RFID instrument
db# bootrom: svn 486-suspect 2011-07-18 12:48:52
db# os: svn 486-suspect 2011-07-18 12:48:57
db# FPGA image built on 2009/12/ 8 at 8: 3:54
```

如果固件版本高于r798，那么版本输出应该是如下这类格式：

```
db# Prox/RFID mark3 RFID instrument
db# bootrom: svn 698 2013-04-17 10:19:38
db# os: svn 0 2014-03-21 08:15:55
db# FPGA image built on 2014/02/25 at 07:43:59
```

```
uC: AT91SAM7S256 Rev B
Embedded Processor: ARM7TDMI
Nonvolatile Program Memory Size: 256K bytes
Second Nonvolatile Program Memory Size: None
Internal SRAM Size: 64K bytes
Architecture Identifier: AT91SAM7Sxx Series
Nonvolatile Program Memory Type: Embedded Flash Memory
```

把天线连接至ProxmarkIII主板，如图2-16所示。



图2-16　连接天线至ProxmarkIII主板

然后在ProxmarkIII客户端的CMD窗口中输入“hw tune”，即可监测天线的工作状况，一般情况下，会得到类似于下面的输出：

```
db# Measuring antenna characteristics, please wait.
LF antenna: 33.17 V @ 125.00 kHz
LF antenna: 41.89 V @ 134.00 kHz
LF optimal: 41.76 V @ 127.66 kHz
HF antenna: 7.28 V @ 13.56 MHz
```

通过这个命令提示符，我们可以操纵ProxmarkIII进行高/低频卡复制、读卡器嗅探、高频卡破解。

但是ProxmarkIII的功能众多，对应的命令和参数也繁多，为了方便客户端使用，所以有人开发出Windows平台下的GUI版本客户端，

名为Proxmark Client，大大减少了ProxmarkIII客户端的使用难度，界面如图2-17所示。

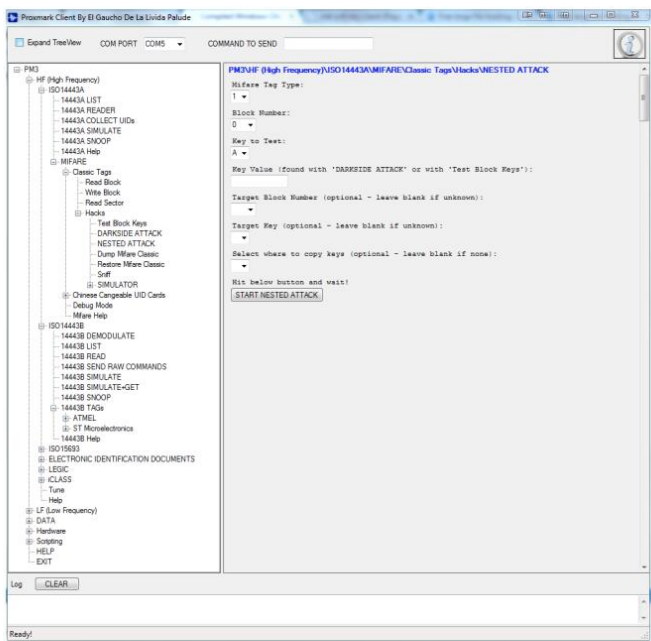


图2-17　Proxmark Client界面

2.3.4　ProxmarkIII安全测试Mifare Classic用例

我们使用命令行版ProxmarkIII客户端进行一次普通的Mifare Classic智能卡安全测试，借此给初学者演示如何使用ProxmarkIII。

首先将天线连接上ProxmarkIII，连接完成后，需要查看天线是否连接正常，以及电压是多少。

在ProxmarkIII客户端的命令提示符后输入如下命令：

```
hw tune
```

等待数秒之后，即可看到命令回显，从回显中可以看到，ProxmarkIII对应天线的当前电压及工作状态，如图2-18所示。

```
proxmark3> hw tune
#db# Measuring antenna characteristics, please wait.

# LF antenna:  0.13 V @   125.00 kHz
# LF antenna:  0.13 V @   134.00 kHz
# LF optimal:  0.00 V @ 12000.00 kHz
# HF antenna:  9.22 V @    13.56 MHz
# Your LF antenna is unusable.
```

图2-18　ProxmarkIII对应天线的当前电压及工作状态

当用户输入“hw tune”命令之后，可以看到高／低频天线当前的电压状态，以及低频天线3个分频率对应的电压。在图2-18中我们可以看到，当前不存在低频天线，而高频天线的电压为9.22V，这个电压是天线的非工作电压。如果把高频卡放置到高频天线上，对应的电压会发生比较大的变化，如图2-19所示。

图2-19所示是我们把一张Mifare Classic卡放置到高频天线上执行“hw tune”命令后的结果，可以看到高频天线的电压从9.22V变化到3.90V。当然，此刻的电压依然属于非工作电压，ProxmarkIII开始工作之后的天线电压才为工作电压，此时电压变化巨大，这是因为天线和高频卡内部的天线耦合工作导致的电压下降。

我们换下高频卡，放上一张125kHz的低频卡，再次执行“hw

tune”命令进行天线电压检查，这时会得到如图2-20所示的电压及工作状态。

```
# LF antenna:   0.00 V @    125.00 kHz
# LF antenna:   0.13 V @    134.00 kHz
# LF optimal:   0.00 V @ 12000.00 kHz
# HF antenna:   3.90 V @     13.56 MHz
# Your LF antenna is unusable.
# Your HF antenna is marginal.
```

图2-19　放置高频卡之后的天线电压

```
# LF antenna:   0.00 V @    125.00 kHz
# LF antenna:   0.00 V @    134.00 kHz
# LF optimal:   0.00 V @ 12000.00 kHz
# HF antenna:   8.57 V @     13.56 MHz
# Your LF antenna is unusable.
```

图2-20　放置低频卡之后的天线电压及工作状态

对比图2-19和图2-20可以轻易地发现很大的不同，放置低频卡之后的电压变化比放置高频卡之后的电压变化要小得多，这个不同之处经常用于识别一张未知的卡是高频卡还是低频卡。

对于当前的Mifare Classic1K卡，我们可以通过执行“hf14a reader”命令来读取卡的基本信息，如图2-21所示。

```
proxmark3> hf 14a reader
ATQA : 04 00
 UID : [redacted] 00 00 00 00
 SAK : 08 [2]
 SAK : MIFARE CLASSIC 1K
proprietary non-iso14443a card
```

图2-21　Proxmark III读取Mifare Classic高频卡的基本信息

从图2-21中的信息我们可以看到，现在读取的卡[ATQA]为0400，ATQA为0400数值的卡，大部分都属于Mifare Classic或者是CPU兼容模式下的Mifare Classic。

确定这张卡的属性之后，就可以深入了解卡的信息并进行破解了。

我们在拿到一张Mifare Classic卡之后，首先要做的就是对卡进行默认Key检测。Mifare Classic系列卡存在的一个主要问题就是，很多卡有些区块的加密密钥是默认密钥，而这些默认密钥很多是ABAB/AABB的形式，我们在之前提到的Mifare Classic的一个漏洞中讲到，Mifare Classic可以通过一个已知密钥推算得出其余所有的Key，因此我们通常会使用如下命令检测卡是否存在已知的默认密钥：

```
hf mf chk * ?
```

命令中的“*”表示检查所有区块，“？”表示使用ProxmarkIII内置的所有Key进行逐个检查，最后会向客户端反馈是否找到可用的Key。使用GUI版本测试的结果如图2-22所示。

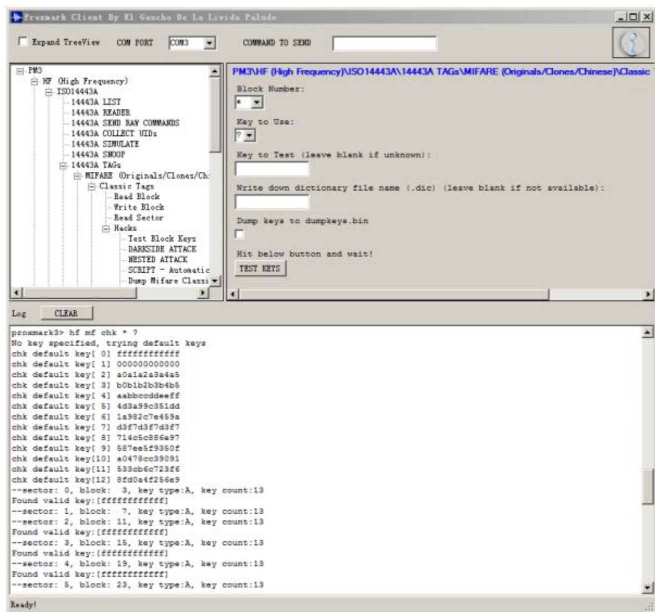


图2-22　查找默认密钥

从图2-22中我们看到，最后找到了一个FFFFFFFF的Key，为有效Key。

在这里为了接下来演示方便，直接使用一个密钥进行测试：

```
hf mf chk 0 A a0a1a2a3a4a5
```

命令回显如图2-23所示，isOK表明密钥a0a1a2a3a4a5是0区的Key A。

```
proxmark3> hf mf chk 0 A a0a1a2a3a4a5
--block no:00 key type:00 key count:1
isOk:01 valid key:a0a1a2a3a4a5
```

图2-23　命令回显

得到一个密钥后，我们就可以利用上文所描述的漏洞来推测出所有区块的Key。但是如果默认密钥列表中没有可用的Key，怎么办？我们就需要用到另外一个漏洞—PRNG漏洞，使用ProxmarkIII利用这个漏洞需要执行如下命令：

```
hf mf Mifare
```

这个过程花费的时间会比较久，如果中途需要取消这个操作，记住不要使用“Ctrl＋C”键，而是使用ProxmarkIII板上的按钮来取消操作；否则可能出现ProxmarkIII没有正常工作的现象，只能重新插拔进行初始化了。在这个过程结束之后，会出现两种类型的结果，一种是invalid key，一种是valid key，只有第二种结果出现时，给出的Key才是正确的Key。

```
proxmark3> hf mf mifare
-------------------------------------------------------------------------------
Executing command. It may take up to 30 min.
Press the key on proxmark3 device to abort proxmark3.
Press the key on the proxmark3 device to abort both proxmark3 and client.
-------------------------------------------------------------------------------
.................................................................................
.................................................................................
.....................................

isOk:01


uid([redacted]) nt([redacted] par([redacted]) ks([redacted])


|diff|<nr>    |ks3|ks3^5|parity         |
+----+--------+---+-----+---------------+
| 00 |00000000| e |  b  |1,1,1,1,1,0,1,1|
| 20 |00000020| 2 |  7  |1,1,1,0,1,0,0,0|
| 40 |00000040| 8 |  d  |1,1,1,0,0,0,#db# COMMAND mifare FINISHED

1,0|
| 60 |00000060| 9 |  c  |1,1,1,1,1,1,1,0|
| 80 |00000080| 7 |  2  |1,1,1,0,0,1,1,0|
| a0 |000000a0| 8 |  d  |1,1,1,1,0,0,0,0|
| c0 |000000c0| 1 |  4  |1,1,1,1,0,0,1,0|
| e0 |000000e0| b |  e  |1,1,1,1,1,0,1,0|
-----------------------------------------------------------------------------
Key found:[redacted]

Found invalid key. ( Nt=[redacted]
```

图2-24　找到invalid Key的ProxmarkIII截图

当ProxmarkIII进行PRNG漏洞攻击之后，如果给出的结果是

invalid key，那么要记住最后给出的Nt值，然后使用这个值进行下一次PRNG漏洞攻击。假如结果如图2-24所示，那么就要执行如下命令来进行下一次操作：

```
hf mf mifare 4b11b6a2
```

输入上一次得出的Nt值之后，重复进行上一次操作。基于PRNG漏洞的攻击是比较耗时的，PRNG漏洞攻击的结果可能会出现多次Nt结果循环，只需要重复这个过程即可，所以尽量采用默认Key攻击的方式来破解。如图2-25所示就是一个PRNG攻击成功结果的典型例子，即发现了valid key。

```
p1:14e p2:4d16 p3:0 key:fd404f4307c7
p1:152 p2:4e76 p3:1 key:fd33ae67809c
p1:336 p2:b7f0 p3:2 key:f9774e94fae1
p1:a9e p2:263e1 p3:3 key:ea1a530edf9f
p1:d7b p2:304de p3:4 key:e45b4813fc68
p1:ef7 p2:357ee p3:5 key:e16bdaf86f34
p1:f95 p2:37856 p3:6 key:e045a4bcdda2
p1:147a p2:484d3 p3:7 key:d6943a5a3492
p1:15d5 p2:4d49c p3:8 key:d3bcac172843
p1:1a30 p2:5bf2b p3:9 key:cb59ea1492a4
p1:1ba6 p2:612f6 p3:a key:c85cb6203c76
p1:20bc p2:73424 p3:b key:be0281f9c778
p1:223a p2:78bff p3:c key:badfd41fe80e
p1:27ec p2:8cca2 p3:d key:af7eee71413f
p1:2910 p2:90384 p3:e key:ad7b5e003602
p1:2984 p2:921d9 p3:f key:ac64a06d47e7
p1:29e7 p2:935be p3:10 key:abadcba4fe35
p1:2dae p2:a00ff p3:11 key:a4638397e6dd
p1:2f2c p2:a55ae p3:12 key:a15eed3ea607
p1:3171 p2:acbf6 p3:13 key:9d207168c4fe
p1:34a8 p2:b7769 p3:14 key:96f7bf0463b6
p1:3794 p2:c1a67 p3:15 key:9120f1917744
p1:3967 p2:c8052 p3:16 key:8d8623018ca6
p1:3bf6 p2:d08dd p3:17 key:889fbacd49e3
p1:3cc9 p2:d38d3 p3:18 key:86e7e8912003
p1:4141 p2:e2461 p3:19 key:7e762f25c444
p1:44cf p2:ee951 p3:1a key:77655de076af
p1:4860 p2:fa189 p3:1b key:70c0f5277a00
p1:4bdc p2:105572 p3:1c key:6a5842a4ff6c
p1:4f7f p2:111fb3 p3:1d key:6322a4920041
p1:548b p2:123a0c p3:1e key:58f61f6dffff
p1:564c p2:129df8 p3:1f key:556b2d8ee386
p1:583a p2:13072a p3:20 key:51aa893e08fb
p1:58cd p2:132978 p3:21 key:507093ff4af6
p1:5b1d p2:13a8c9 p3:22 key:4be52c45ce75
p1:5da1 p2:1435d1 p3:23 key:46ca5924492c
p1:691e p2:16b6ba p3:24 key:2fd725259c52
p1:6a44 p2:16f8b6 p3:25 key:2d73f745ebfb
p1:6b74 p2:17394b p3:26 key:2b233693dc5d
p1:6f29 p2:180916 p3:27 key:23b0c09f38c0
p1:71f4 p2:189591 p3:28 key:1ea7b85b5240
p1:7278 p2:18b066 p3:29 key:1db30198d85d
p1:7606 p2:197d2a p3:2a key:165b062041dc
p1:76d1 p2:19a642 p3:2b key:14e71dfaf558
p1:7810 p2:19e9cf p3:2c key:1279abdca2f5
p1:7989 p2:1a43d9 p3:2d key:0f40149d93f9
p1:799f p2:1a4741 p3:2e key:0f20677dc36d
p1:7ede p2:1b6980 p3:2f key:04bd3293eef9
key_count:48
--------------------------------------------------------------------
Key found:badfd41fe80e

Found valid key:badfd41fe80e
proxmark3>
```

图2-25　PRNG攻击成功截图

当我们采用各种方法获得了一个正确的Key之后，就可以根据这个Key来获取全部区域的Key值了。在ProxmarkIII中，这种攻击方式又称为NESTED攻击。以我们最先获得的Key Aa0a1a2a3a4a5为例，执行如下命令：

```
hf mf nested 1 0 A a0a1a2a3a4a5
```

即可对卡进行NESTED攻击，攻击结果类似于图2-26。

```
Iterations count: 25
|---|----------------|---|----------------|---|
|sec|key A           |res|key B           |res|
|---|----------------|---|----------------|---|
|000|  a[redacted]a1a2a3[redacted]4a5  | 1 |  f9[redacted]257[redacted]bdf[redacted]3  | 1 |
|001|  e[redacted]9bd361[redacted]01b  | 1 |  60[redacted]345[redacted]d37[redacted]a  | 1 |
|002|  a[redacted]ce2b28[redacted]013  | 1 |  45[redacted]806[redacted]768[redacted]b  | 1 |
|003|  8[redacted]4dcca6[redacted]0bf  | 1 |  59[redacted]80b[redacted]d4f[redacted]3  | 1 |
|004|  6[redacted]bff592[redacted]e7d  | 1 |  f9[redacted]257[redacted]bdf[redacted]3  | 1 |
|005|  e[redacted]9bd361[redacted]01b  | 1 |  f9[redacted]257[redacted]4bdf[redacted]3  | 1 |
|006|  e[redacted]9bd361[redacted]01b  | 1 |  f9[redacted]25[redacted]4bdf[redacted]3  | 1 |
|007|  7[redacted]046f25[redacted]053  | 1 |  60[redacted]34[redacted]5d370[redacted]  | 1 |
|008|  d[redacted]9fa5c[redacted]946  | 1 |  b0[redacted]1b[redacted]b3b4[redacted]  | 1 |
|009|  a[redacted]a1a2a[redacted]4a5  | 1 |  b0[redacted]1b2[redacted]3b4[redacted]5  | 1 |
|010|  a[redacted]a1a2a3[redacted]4a5  | 1 |  b0[redacted]1b2[redacted]3b4[redacted]5  | 1 |
|011|  a[redacted]a1a2a3[redacted]4a5  | 1 |  b0[redacted]1b2[redacted]3b[redacted]b5  | 1 |
|012|  e[redacted]9bd361[redacted]01b  | 1 |  f9[redacted]257[redacted]bd[redacted]33  | 1 |
|013|  e[redacted]9bd361[redacted]01b  | 1 |  f9[redacted]257[redacted]bd[redacted]3  | 1 |
|014|  e[redacted]9bd361[redacted]01b  | 1 |  f9[redacted]2574[redacted]df[redacted]3  | 1 |
|015|  e[redacted]9bd361[redacted]01b  | 1 |  f9[redacted]574[redacted]df[redacted]3  | 1 |
|---|----------------|---|----------------|---|
```

图2-26　NESTED攻击结果截图

这时我们找出了Mifare Classic卡所有区块的密钥，即可对卡进行修改、复制等后续操作了。

在这里我们要对卡进行复制，就要使用Proxmark Client的脚本进行一个非常简单的连续化操作，即使用Automatic Mifare Crack Script进行自动化爆破，如图2-27所示。

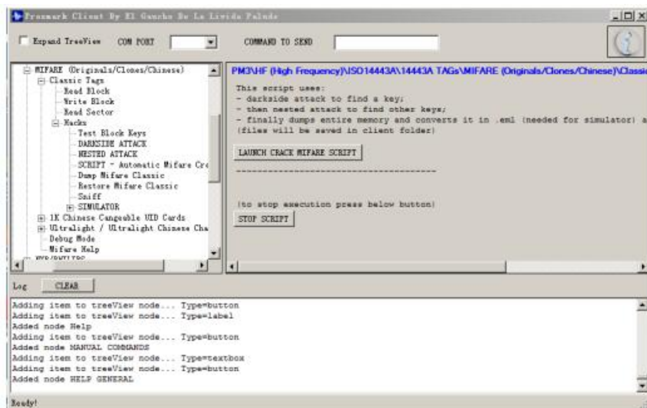


图2-27　使用Automatic Mifare Crack Script

在成功破解出所有的Key之后，使用NESTED中的d选项即可将所有的Key另存为DUMPKEY.bin，然后利用所有已知的Key将Mifare Classic的内容全部导出。

2.3.5 Chameleon-Mini简介

Chameleon-Mini又称变色龙，是德国人开发的一款用于Mifare系列卡模拟的硬件产品，将获得的Mifare系列智能卡的Dump内容导入之后，Chameleon-Mini即可在遇到读卡器时，直接采用智能卡与读卡器交互的逻辑进行工作，从而模拟出一张智能卡。

Chameleon-Mini实物图如图2-28和图2-29所示。



图2-28　Chameleon-Mini早期版本实物图



图2-29　Chameleon-Mini近期版本实物图

2.3.6　Chameleon-Mini固件烧写及使用

将Chameleon-Mini在GitHub上的工程克隆至本地之后，我们需要如下几个设备对Chameleon-Mini进行固件烧写工作。

- Windows7专业版x64PC
- USB Mini-A线
- AVRISP mkII
- Chameleon-Mini

Chameleon-Mini是基于AVR单片机的一款硬件产品，所以对它进行烧写还需要用Atmel Studio进行固件编译和烧写，到AVR的官网下载Atmel Studio6.2，安装完成后打开。

连接AVRISP mkII、PC、Chameleon-Mini，注意6根连接线的防插反端子的凸起对应着Chameleon-Mini的背面（即无贴片元件的一面），连接方式如图2-30所示。



图2-30　Chameleon-Mini烧写端子连接

在Chameleon-Mini连接到AVRISP mkII之后，将Chameleon-Mini连接上电源进行供电，确保Chameleon-Mini以及AVRISP mkII共计3个绿灯全部亮起，如图2-31所示。



图2-31　Chameleon-Mini连接正常

依次单击AVR Studio的“Tools”→“Device Programming”，“Tool”选择“AVRISP mkII”，“Device”选择“ATxmega32A4U”，“Interface”选择“PDI”，单击“Apply”按钮，如图2-32所示。

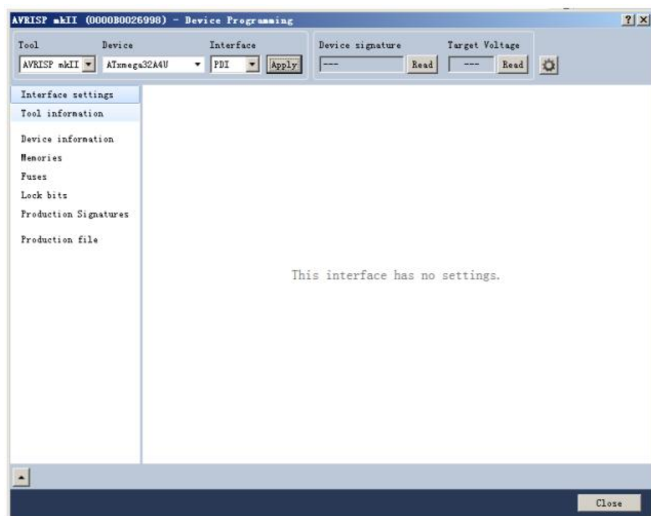


图2-32　AVRISP型号选择

选中“Memories”，选择“Device”→“Erase Chip”，单击“Erase now”按钮，清除原有片上的固件。

选中“Fuses”，依次填入：

- FUSEBYTES1-＞gt;0x00
- FUSEBYTES2-＞gt;0xBE
- FUSEBYTES4-＞gt;0xFF
- FUSEBYTES5-＞gt;0xE9

单击“Program”按钮，烧写AVR单片机的Fuse位，如图2-33所示。

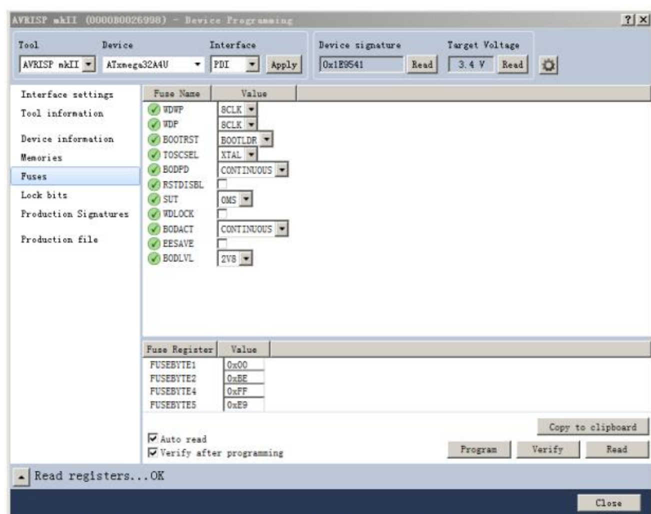


图2-33　AVRISP Fuse位烧写

选中“Production file”，在“Program device from ELF prodution file”栏中，选择“YourPath\ChameleonMini-master\Firmware\Chameleon-Mini\Chameleon-Mini.elf”文件，勾选“Flash”和“EEPROM”，单击“Program”按钮，如图2-34所示。

拔掉AVRISP mkII，重新插拔Chameleon-Mini，这时可能会出现硬件没有相应驱动的情况。同样，更新驱动，选中克隆到本地的官方文件的driver文件夹即可。

打开串口通信软件，这里推荐使用XShell或者SecureCRT，官方推荐使用的是Tera Term，读者可以根据自己的喜好来选择使用。这里需要说明的是，由于Chameleon-Mini使用的是LUFA模拟串口驱动，所以常见的串口通信工具在使用时基本都是没有回显的，也就是输入了什么命令，读者在PC端是无法看到的，只能盲打。

这时读者可以使用官方推荐的Tera Term软件，连接到串口之后，勾选“设置”→“终端”→“本地回显”，然后确定即可显示输入内容，但是输入错误是没法删除修改的；或者使用SecureCRT，连接到串口之后，勾选“查看”→“交互窗口”，然后在下面的交互窗口中输入命令后回车即可。

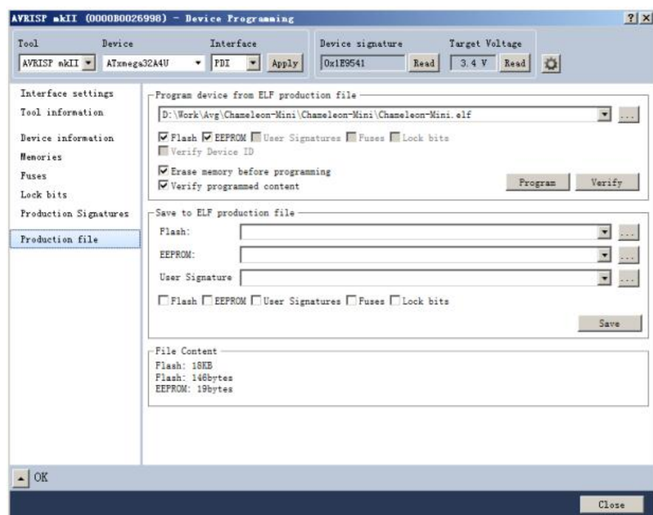


图2-34　选择烧写固件文件

如图2-35所示是Chameleon-Mini正常使用时串口交互的截图。

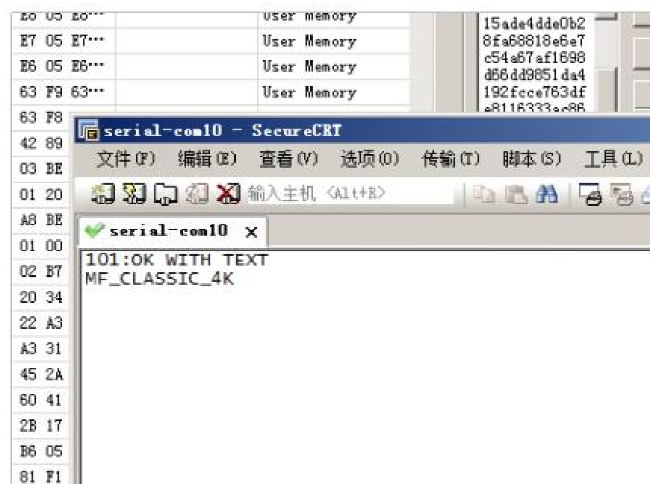


图2-35　正常使用Chameleon-Mini时的串口交互截图

2.3.7 ProxmarkIII与Chameleon-Mini配合模拟Mifare Classic

现在我们有了两个“大杀器”用来对付Mifare系列智能卡，其中一个是ProxmarkIII，可以利用各种漏洞组合破解Mifare Classic智能卡，得到智能卡所有区块的密钥，从而Dump出卡内内容；另一个是Chameleon-Mini，可以使用前面ProxmarkIIIDump出的卡内内容与读卡器进行模拟交互，从而模拟出一张一模一样的Mifare Classic智能卡。

Chameleon-Mini是根据所上传的Mifare Dump文件来模拟出一张一模一样的13MHz Mifare标准卡的，所以需要ProxmarkIII来首先Dump出所要复制的卡文件。

Chameleon-Mini现在只支持模拟Mifare Classic1K卡和4K卡，但是在源代码中可以发现作者预留了Mifare Ultralight、ISO15693_GEN、ISO14443A_SNIFF、ISO15693_SNIFF的接口，鉴于作者的更新频率和已经逐步实现的功能，我们有理由相信这些功能会在不久的将来问世。

接下来我们就介绍如何使用ProxmarkIII和Chameleon-Mini模拟出一模一样的Mifare Classic1K智能卡。

首先将ProxmarkIII连接至计算机，打开Proxmark Client，连接至串口之后，打开“PM3”→“HF”→“ISO14443A”→“14443A Tags”→“MIFARA”→“Classic Tags”→“Hacks”→“SCRIPT”→“Automatic Mifare Crack”，把要破解的卡放到HF天线上，单击“LAUNCH CRACK MIFARE SCRIPT”按钮，等到最后出现：

```
Dumped 64 blocks (1024 bytes) to file dumpdata.bin
Wrote a HTML dump to the file 7778863C.html
Wrote an emulator-dump to the file 7778863C.eml
```

时，说明卡的内容已经Dump到7778863C.eml中了。

但是eml格式的数据文件不能直接在Chameleon-Mini中使用，所

以要进行转换。打开Windows CMD窗口，执行“cd C：\Proxmark\ProxSpace\pm3\client”，然后执行“pm3_eml2mfd.py7778863C.eml7778863C.mfd”，这样，Chameleon-Mini要用的mdf文件就已经转换生成了。

打开SecureCRT，连接至串口，输入“UPLOAD”命令，打开“传输”→“发送Xmodem”，找到刚才生成的7778863C.mfd文件，传送成功后根据Dump出的卡的型号执行“CONFIG＝MF_CLASSIC_1K”或者“CONFIG＝MF_CLASSIC_4K”命令，这里我们模拟的是一张Mifare Classic1K卡，所以使用的是“CONFIG＝MF_CLASSIC_1K”命令。

读者可以查看“BUTTON”命令的使用方法，设置之后用户可以操作模拟卡的UID值，按动按键，左/右位递增/递减UID值，或者随机生成UID值都可以。

2.3.8 RFID高频攻防总结

到这里，我们针对RFID中的一大用例—Mifare Classic智能卡的破解讲解就暂告一段落了，Mifare Classic智能卡之前在大型的公共交通系统中使用非常广泛，因此在早先Mifare Classic爆出重大安全漏洞之后的市场影响也非常巨大，北京的交通一卡通最早采用的也是Mifare Classic4K卡的解决方案，从漏洞评估到最后全面更换，消耗了巨大的人力、物力资源。

这种破解方式，现如今在使用Mifare Classic智能卡作为解决方案的场合，依然大部分情况都可行。之所以说大部分，是因为NXP半导体公司早已推出新一代的基于Mifare Classic并且进行安全加固后的智能卡解决方案，虽然统一被识别为Mifare Classic智能卡，但是ProxmarkIII依赖的漏洞早已不再能攻击成功。建议依然采用包含漏洞的Mifare Classic智能卡的场所及时更换解决方案及硬件设施，避免造成不必要的损失。

2.4 低频ID卡安全分析

低频非接触式ID卡主要应用于门禁、考勤等系统，使用很广泛，但是具有比较大的安全隐患，由于没有密钥安全认证的相关机制，所以只要对低频ID卡的编码/解码机制有所研究，就能够很容易地对这种类型的卡进行破解和复制。

本节将对125kHz射频ID卡进行分析和信息读取，从而带领读者对低频ID卡的安全性有进一步的了解。

2.4.1　低频ID卡简介

低频（Low Frequency，LF）是指频段由30kHz到300kHz的无线电电波。一些无线电频率识别（RFID技术）标签使用低频，这些标签通常被称为LFID's或LowFID's（Low Frequency Identification，低频率识别）。然而，LFID's/LowFID's所常用（非唯一）的频率为125kHz/134kHz，125kHz/134kHz只是低频RFID所基于的频率，该频率不存在任何功能性，就是说频率本身不存在所谓的ID识别、读取/写入等。

非接触式125kHz ID卡主要应用于门禁卡、考勤卡、企业一卡通等领域，由于该芯片出厂固化了全球唯一的一组64位的编码，类似于身份证号码，采用的是感应式读卡方式，读卡速度快，不受卡面污染影响；一般管理系统都会使用，具有操作方便、快捷、可靠、寿命长等优点；主要应用于食堂售饭系统、巡更系统、门禁系统、企业一卡通之类的射频识别领域，在我们身边最常见的就是门禁卡。常见的低频ID卡有HID、T55xx、EM410x等系列。

如图2-36和图2-37所示即为常见的低频ID卡的实体形式，一种为卡片式，一种为纽扣式。



图2-36　纽扣式低频ID门禁卡

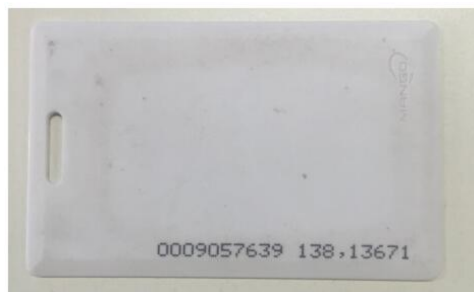


图2-37　卡片式低频ID门禁卡

2.4.2 ID卡编码原理

125kHz ID卡的编码方式采用曼彻斯特编码（Manchester Encoding）。曼彻斯特编码也叫作相位编码（Phase Encode，PE），是一种同步时钟编码技术，被物理层使用来编码同步位流的时钟和数据。

在编码位的1/2处，若被编码数据位为“1”，则为负跳，反之为正跳；在编码位的开始处，若被编码数据位为“1”，则为高电平，反之为低电平。由于曼彻斯特编码每一个码元的正中间会出现一次电平的转换，这对接收端的提取位同步信号是非常有利的。虽然曼彻斯特编码需要比较复杂的技术，但是可以获得比较好的抗干扰特性。

2.4.3 ID卡译码原理

根据曼彻斯特的编码原则，非接触式ID卡采用上升沿对应着位数据“0”，下降沿对应着位数据“1”，微控制器通过监测读卡头输出数据位的跳变来实现对曼彻斯特的译码。在实际应用中，数据信号会受到调制、解调、噪声等各种效应的影响，其上升沿和下降沿会存在抖动，可采用键盘消抖的方式来消除抖动的影响。

在工作状态下，只要射频基站电路不断点，非接触式ID卡就会循环发送64位数据，数据结构如图2-38所示。

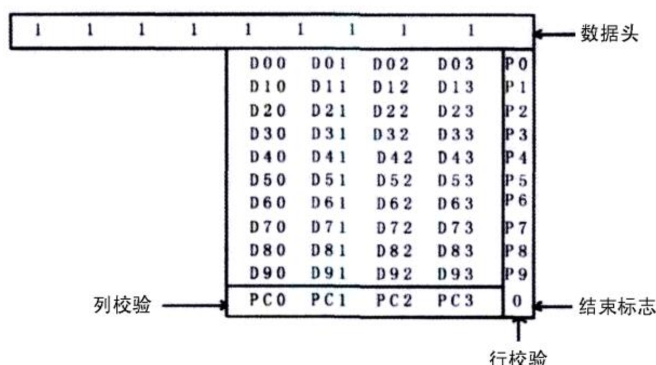


图2-38　ID卡64位数据结构

其中，连续9位“1”作为头数据，是读取数据时的同步标识；D00～D93是用户定义数据位；P0～P9是行奇校验位，PC0～PC3是列奇校验位；最后“0”是结束标识。

鉴于上述曼彻斯特码的数据结构，我们可以将“0111111111”作为数据的起始标识。在确定了数据起始标识后，采用延时大于0.5T采样数据位的方法，去除曼彻斯特码中的空跳对数据译码造成的影响，简化译码程序。

2.4.4 ID卡数据读取

ID卡的内容可以直接用读卡器读出，如图2-39所示。



图2-39 ID卡读卡器

使用ID卡读卡器时直接将其通过USB接口插到计算机上，然后打开记事本，Windows内部驱动会直接将读卡器的读取数据显示在记事本上。假如我们将一张低频测试卡放置到工作中的ID卡读卡器上，记事本的显示结果如图2-40所示。

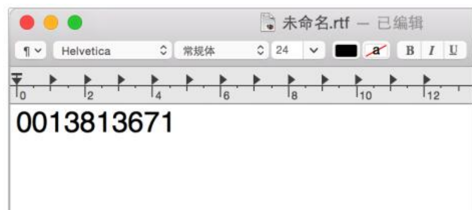


图2-40 ID卡读卡器的读取结果

2.4.5　ID卡卡号格式

由于各个厂家的ID卡卡号读卡器的译码格式不尽相同，在读卡输出时，读出的二进制或十六进制（Hex）结果应该是唯一的，但是可以通过以下几种主要换算办法，输出不同结果的十进制卡号（DEC）。

（1）ID卡卡号格式0：10位十六进制的ASCII字符串，即10Hex格式。如某样卡读出十六进制卡号为“01026f6c3a”。

（2）ID卡卡号格式1：将格式1中的后8位转换为10位十进制卡号，即8H—10D。如将“026f6c3a”转换为“0040856634”。

（3）ID卡卡号格式2：将格式1中的后6位转换为8位十进制卡号，即6H—8D。如将“6f6c3a”转换为“07302202”。

（4）ID卡卡号格式3：将格式1中的倒数第5、6位转换为3位十进制卡号，再将后4位转换为5位十进制卡号，中间用“，”分开，即“2H＋4H”。如将2H“6f”转换为“111”，4H“6c3a”转换为“27706”，最终将两段号连在一起输出为“111，27706”。

（5）ID卡卡号格式4：将格式1中后8位的前4位转换为5位十进制卡号，再将后4位转换为5位十进制卡号，中间用“，”分开，即“4Hex＋4Hec”。照此推算，结果为“00623，27706(4H＋4H)”。

我们对一张卡面上标有两种输出格式的低频ID卡厚卡进行读取，在读取过程中改变读卡器的输出格式，然后与卡面上的标号进行对照，如图2-41所示。

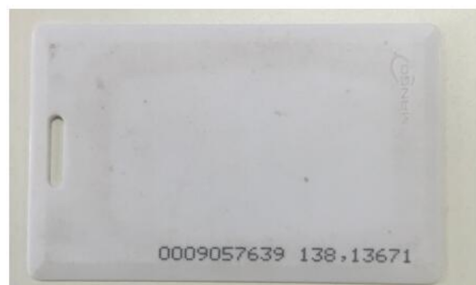


图2-41　有两种格式标识的低频ID卡

改变读卡器的输出格式，得到两种不同的读卡器输出内容，对比发现，输出内容与卡面上的标识完全一致，如图2-42所示。



图2-42　读卡器两种格式的输出内容

2.5 低频ID卡克隆攻击

从上面对低频ID卡工作方式及工作机理的描述中可以得出这样一个结论，我们用来举例的常见低频ID卡的卡内内容可以直接被任意读卡器读取，卡内数据未经过加密，能起到身份标识作用的就是卡内固化的64位数据。而一般的低频ID卡在出厂前，会烧断熔丝位，避免卡内数据在出厂后被修改。但是即使这些卡的熔丝位不被烧断，也不会影响卡的正常使用，这也就催生了另一个行业—低频ID卡白卡，这种卡的熔丝位在出厂前不被烧断而直接售卖，这样购买的用户有一套写卡器即可将卡内数据修改为任意数值，从而做到克隆攻击。很多配钥匙的地方可以“克隆门禁卡”也是利用这种漏洞，配钥匙的人员对低频ID卡使用普通读卡器读出卡内明文数据，然后将数据写入白卡内，即做到了“门禁卡克隆”。

在这里我们暂时不介绍这种简单的方式，而是介绍另外几种更为万能和多用的克隆攻击方式。

下面以ID卡门禁系统为例，给大家说说这几种攻击方式。ID卡门禁系统是比较常见的，其安装方便、成本低、使用方便，很多中小写字楼、小区都使用该系统。常见的门禁控制器如图2-43所示。



图2-43　常见的门禁控制器

这样的门禁系统一般主要有3部分组成：门禁控制器、门禁卡和锁系统。其原理是，门禁控制器中存储了可开锁的门禁卡数据（相当于白名单），当使门禁卡靠近门禁控制器时，门禁控制器中的读卡模块读取门禁卡数据，然后查询白名单，如果卡数据在白名单中则打开锁系统；如果遍历白名单后没有查询到结果则关闭锁系统。

常见的ID门禁卡从外形上看有3种：薄卡、厚卡和纽扣卡，外形对比如图2-44所示。



图2-44　常见的ID门禁卡对比

2.5.1　ProxmarkIII模拟攻击

要进行攻击的首要条件就是先获得ID卡里面的数据信息。我们既可以用之前所介绍的低频ID卡读卡器，也可以使用ProxmarkIII的低频功能进行卡内信息读取。在这里我们用一张EM4100种类的低频ID卡来演示，打开ProxmarkIII客户端，执行如下命令读取卡内信息：

```
lf em4x em410xwatch
```

得到如图2-45所示的结果。

从图2-45中我们可以看到这张卡的卡号是030031fb2a，接下来就可以利用ProxmarkIII的模拟功能进行低频ID卡模拟，根据前面得到的卡内数据，执行如下命令：

```
lf em4x em410xsim 030031fb2a
```

这时ProxmarkIII就开始向外部发送该卡号的数据，如图2-46所示。可以看到ProxmarkIII的指示灯颜色为黄色，即为数据模拟发送阶段。

```
C:\WINDOWS\system32\cmd.exe - proxmark3.exe
proxmark3> lf em4x em410xwatch
#db# buffer samples: 00 00 00 00 00 00 00 00 ...
Reading 2000 samples

Done!

Auto-detected clock rate: 64
Thought we had a valid tag but failed at word 1 (i=45)
Thought we had a valid tag but failed at word 2 (i=114)
#db# buffer samples: 9d 7c 60 5b 38 3a 1e 14 ...
Reading 2000 samples

Done!

Auto-detected clock rate: 64
EM410x Tag ID: 030031fb2a
proxmark3> _
```

图2-45　ProxmarkIII读取低频卡信息

```
proxmark3> lf em4x
help              This help
em410xread        [clock rate] -- Extract ID from EM410x tag
em410xsim         <UID> -- Simulate EM410x tag
em410xwatch       Watches for EM410x tags
em4x50read        Extract data from EM4x50 tag
proxmark3> lf em4x em410xsim 030031fb2a
Sending data, please wait...
Starting simulator...
proxmark3>
```

图2-46　ProxmarkIII模拟低频ID卡

2.5.2 白卡克隆攻击

读者应该清楚ProxmarkIII的体积和需要上位机配合的特点，在研究中使用信号模拟的方法是比较方便的，但是在实际运用中就不是很方便了。试想你在众目睽睽之下，拿着单片机和天线在门禁上晃悠，那会是怎样的景象啊？所以，在获得了ID卡内数据后，完全可以伪造一张卡。

使用的工具有：低频读写器和可写的ID卡。当然，你也可以继续使用ProxmarkIII来进行写卡操作，不过在实践中我们发现ProxmarkIII的低频功能不是很稳定，所以推荐大家使用专门的低频读写设备，如图2-47和图2-48所示。

低频卡的读卡/写卡都是非常方便的，只需要将读出的卡号输入，既可完成写卡，如图2-49所示。



图2-47 读卡操作



图2-48　写卡操作



图2-49　读卡/写卡配套软件

2.5.3 HackID模拟攻击

HackID是360UnicornTeam无线电硬件安全实验室制作的一款针对低频ID卡进行模拟攻击的硬件工具，功能强大，包含有键盘、中控芯片、125kHz天线，可以做到在输入序列号之后，硬件即开始自动模拟出对应的低频ID卡，可持续使用，小巧并可随身携带。HackID实物图如图2-50所示。

HackID使用时非常简单、方便，打开电源，输入读取到的低频ID卡序列号，确定之后，设备将会循环发送ID号对应的低频信号，将设备贴近读卡器即可开始正常工作。



图2-50 HackID实物图

2.6 EMV隐私泄露

2.6.1 EMV简介

EMV，命名来自于Europay、万事达卡（MasterCard）与威士卡（VISA）三大国际组织英文名称的字首。EMV是国际金融业界对于智能卡与可使用芯片卡的POS终端机，以及银行机构所广泛设置的自动柜员机等所制定的专业交易与认证的标准规范，是针对芯片信用卡与现金卡（借记卡）的支付款系统（Payment System）相关软硬件所设置的标准。

“芯片卡”或可读取芯片卡的终端机与柜员机的规格合乎EMV标准并经过认证无误时，称之为EMV芯片卡，简称EMV卡。这种卡一般支持非接触式支付，对应的非接触式支付品牌有三个：VISA组织的payWave、MasterCard组织的PayPass、UnionPay组织的QuickPass。

虽然中国银联在2013年5月加入了EMV组织，但是实际上中国境内能使用的芯片卡遵循的都不是EMV标准，而是根据中国特色修改后的PBOC标准，这两种协议在物理级别上是相同的，都是CPU卡，都支持APDU（应用协议数据单元）协议；区别是在应用级别。

目前中国发行的芯片卡分为两个标准：EMV2000和PBOC2。其中EMV2000是EMV组织制定的，在中国受限，在其他地方通用；PBOC是UnionPay组织制定的，在中国可用，在其他地方不能用。

而这里要介绍的就是EMV标准中的非接触式支付的个人隐私泄露点，对应我们身边的事物即为含有“QuickPass”标志的银联芯片卡。注意图2-51所示银行卡中左侧的芯片和右侧的“QuickPass”标志。



图2-51　带有QuickPass功能的EMV芯片卡

2.6.2　非接触式芯片卡隐私泄露原理

如果你的手机带NFC功能，并且打开支付宝，将带有“QuickPass”标志的芯片卡贴置手机的背面，则会看到可以读取出手机的银行卡号、姓名、交易记录等敏感信息，如图2-52所示。不过，支付宝的读取信息会在终端上面隐去部分信息，但是实际上在银行卡内部是存储着全部信息的。



图2-52　支付宝读取非接触式芯片卡内容结果

使用安装有支付宝钱包的NFC手机，不同的银行卡可以读取的信息不尽相同，测试结果如表2-1所示。

表2-1 测试结果

银行卡	卡号显示	卡内余额	电子钱包余额	交易记录	身份证号
广州银行	后 4 位	不可读	可读	可读	不可读
建设银行	全卡号	不可读	可读	可读	开头、末尾两位
交通银行	后 4 位	不可读	可读	可读	不可读
招商银行	全卡号	不可读	可读	可读	开头、末尾两位
中国银行	全卡号	不可读	可读	可读	不可读
工商银行	全卡号	不可读	可读	可读	不可读
农业银行	全卡号	不可读	可读	可读	不可读

笔者进行了多家银行数据测试，发现部分银行卡是无法正常识别的。经过反复修改数据发现，支付宝钱包是通过读取0201DGI中的数据识别银行卡号的，发送00B2011444指令，只要0201的长度不等于44H，多数卡片就都会返回6CXX，即无法正确识别卡片。但这不符合PBOC3.0规范要求，PBOC3.0规范第五部分B.12读记录C-APDU/R-APPDU中要求le为00。

那么这些信息是如何实现的呢？首先看银行卡号，银行卡号在芯片卡内以5A标签存在，一般会存放在0201DGI中，发送00B2011400指令即可读出这条以70模板开始的记录数据，根据TLV模板规则解析，即可得到如表2-2所示的数据。

表2-2　00B2011400指令响应解析结果

标签	定义	长度	数据
5F24	应用失效日期	03	241231
5F25	应用生效日期	03	150622
5A	应用主账号	0A	6230910299000378541F
9F07	应用使用控制	02	FF00
8E	CVM 列表，借贷记	0E	0000000000000000042031E031F00
9F0D	IAC 默认借贷记	05	D8609CA800
9F0E	IAC 拒绝借贷记	05	0010000000
9F0F	IAC 联机借贷记	05	D8689CF800
5F28	发卡行国家代码	02	0156
9F08	应用版本号	02	0030

同理，身份证号9F61标签也可以读取到，一般写入到DGI0102中，可以通过00B2020C00读取，这条记录会返回证件号9F61、姓名5F20、证件类型9F62等。

电子现金余额在卡内用9F79标签标识，长度为6字节，目前最大值为1000元即000000100000，由卡内数据9F77（电子现金余额上限）限制，单笔交易最大额度由9F78（电子现金单笔交易限额）限制。当电子钱包余额小于9F6D（电子现金重置阈值）时，卡会自动从主账户圈存金额至9F79。这里介绍的这些数据都可以通过GET DATA指令获取，例如在非接触下发送80CA9F7900，即可获取电子现金余额。但卡内余额是不能够获取的，只有输入联机PIN码，在联机情况

下才可以。

交易日志格式在个人化数据中用9F4F标签标识，在银联模板中推荐了多组值，例如此值为9A039F21039F02069F03069F1A025F2A029F4E149C019F3602，格式为Tag＋Length，表示记录日志中应包含：交易日期（9A）、交易时间（9F21）、授权金额（9F02）、其他金额（9F03）、终端国家代码（9F1A）、交易货币代码（5F2A）、商户名称（9F4E）、交易类型（9C）、应用交易计数器（9F36）。此值由银行根据需要自定义。终端可以通过取数据（GET DATA）命令获取9F4F的值，在非接触下发送80CA9F4F00即可。

那么如何读出记录呢？终端会通过发送读指令获取日志，记录的读取权限为自由。例如发送00B2015C00指令，获取第一条记录，根据PBOC3.0规范第五部分B.12中的定义，前两个字节00B2固定，分别为CLA、INS。第三个字节01表示记录号，也就是需要读取第几条记录。5C为读取的记录文件标识，在9F4D中定义为0B，那么0B*8＋4＝5C。例如读取第一条记录返回
1407080955150000000100000000000000000156015663686 96E61756E69
根据上面所列的标签解析得到如表2-3所示的数据。

表2-3　00B2015C00指令响应解析结果

标签	定义	数据
9A	交易日期	140708
9F21	交易时间	095515
9F02	授权金额	000000010000
9F03	其他金额	000000000000
9F1A	终端国家代码	0156
5F2A	交易货币代码	0156
9F4E	商户名称	6368696E61756E696F6E7061792E616263643132
9C	交易类型	01
9F36	应用交易计数器	0003

表2-3中所显示的内容完整性、数据的多少都是可以由NFC手机软件操控的。金融IC卡中写入的大部分数据都可以自由读取，但不用担心卡会被复制，因为目前的金融IC卡中82（应用交互特征）都支持DDA（动态认证），在动态认证中需要卡的私钥参与认证，私钥是不能被读取的，所以卡不会被完整地克隆。

金融IC卡为了交易方便、快捷，大部分银行在接触电子现金及非接触QPBOC环境下，都选择免输脱机PIN，只需签名或免签就可以完成交易。持卡人验证方法是由卡内数据8E决定的。

终上所述，如果芯片银行卡被未经授权的人使用NFC手机读取了信息，则可能导致银行卡号、证件号码、姓名等相关隐私信息被窃取，但这些信息并不能直接导致银行卡资金被盗。目前BCTC送检已经建议银行在个人化数据中尽量避免出现姓名和证件号码等内容，这些信息可以由银行后台去关联。

2.6.3 非接触式芯片卡隐私泄露现象

上一节我们介绍了非接触式芯片卡个人隐私读取的原理，但是比较晦涩难懂，也无法直观地表现这种安全隐患的影响，为此360UnicornTeam的安全研究人员制作了两款小软件，分别用于Android和Windows平台下的非接触式芯片卡的个人信息读取。

如图2-53所示为Android客户端的读取方式，我们可以看到卡持有人的姓名、银行卡号、身份证号、发卡日期、失效日期以及最近10笔交易记录等个人敏感信息。



图2-53　Android客户端读取非接触式银行卡个人信息

而Windows客户端则采用高频读卡器的方式读取银行卡信息，其读取范围和读取速度比手机端要高出不少，读取银行卡信息如图2-54所示。



图2-54　Windows客户端读取非接触式银行卡信息

2.6.4 非接触式芯片卡个人隐私保护

这里要提及的一点就是，这些非接触式芯片卡内存储的个人信息的多少、敏感程度是由发卡行决定的，并非由银联强制规定。而芯片卡的大量发行，也造成了实际上用户手中的卡泄露了多少个人信息是无法得知的。

而我们前面讲过，这种非接触式芯片卡是通过NFC近场通信的方式与读卡器进行数据交互的，所以可以通过阻断高频信号的方式来保护个人信息安全。

在美国，一般使用的是屏蔽式钱包，钱包内衬使用的是电磁屏蔽材料，这样可以阻断钱包内部的物体与外界进行无线通信交换。但是这样的钱包一般是壳状的，并且体积大、不美观，一般会在角落处标注RFID标志，如图2-55所示。



图2-55 RFID保护钱包

为了更加方便、美观、实用性更高，360UnicornTeam的安全研究人员采用电磁屏蔽材料制作了一种卡套，平时将非接触式银行卡直接插入卡套内，即可隔绝所有的读取请求，保障用户个人信息安全，如图2-56所示。



图2-56　RFID安全保护卡套

第3章　短距离无线遥控系统

生活中，通过遥控信号来控制的产品无处不在，例如遥控灯、遥控窗帘、无线遥控玩具、遥控车库门及停车场的遥控升降杆。这些遥控模块的工作频率一般为433MHz或者315MHz，发射功率小，距离适中。在这两个频段，有很多低成本的收发芯片，技术非常成熟。

除了遥控装置以外，还有一些物联网传感器或者智能家居硬件，也使用这两个频段的射频芯片，例如烟雾报警器、有毒气体泄漏的传感器等。

然而，这种常见的无线通信装置是否安全呢？这一章我们就将分析短距离小发射功率的无线遥控系统的安全性。

3.1　遥控信号嗅探与安全分析

下面以一个停车场的遥控杆为例来进行分析（见图3-1）。

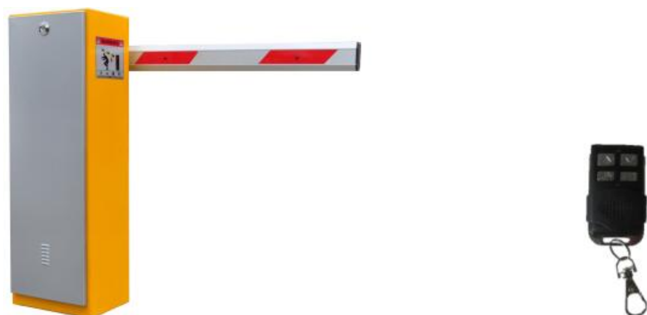


图3-1　停车杆与遥控

首先，对遥控信号进行采集。在这里，我们使用的工具是HackRF和USRP B210。这两种工具的作用是一样的，都能够采集遥控信号，也可以发射信号，可以在433MHz和315MHz频段发射任意波形的信号。先对遥控信号的不同指令进行采集，对停车场的停车杆落下与升起时的控制信号，分别对其采样做若干样本。我们用频谱仪程序，观察信号所处的频段，发现这个停车杆是工作在433MHz频段的。于是，我们使用8MS/s采样率，对中心频率为433MHz的信号采样（准确的中心频率为433.935MHz）。

然后，对采样得到的信号通过Matlab进行分析。如图3-2所示为采样信号的时序分析图，通过该信号复包络的实部可知，遥控信号采用的是ASK/OOK调制与脉冲调制相结合的方法。其中，ASK/OOK是利用载波的幅度变化来传递数字信息的。在OOK中，载波的幅度只有两种变化状态，分别对应着二进制信息“0”或“1”。脉冲调制，即在频率固定的情况下，脉冲的宽度、幅度或相位等任一项按照一定的规律随时间变化而变化的过程，最终达到传输信息的目的。

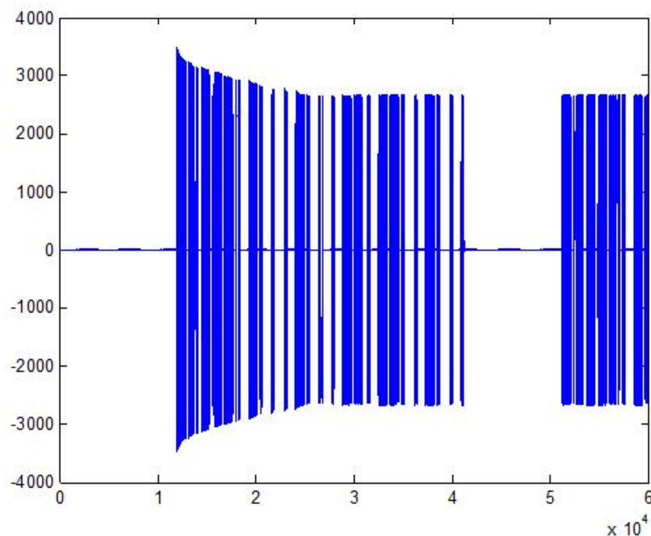
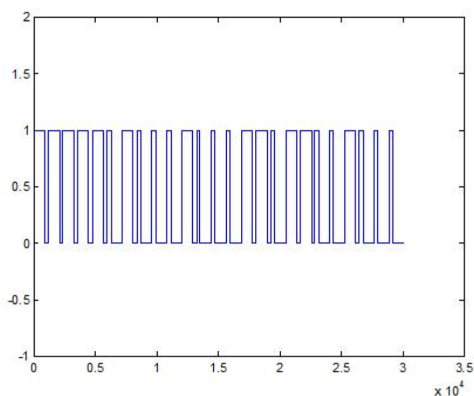


图3-2　采样信号的时序分析图

接下来，对该信号进行解调。采用非相干解调方式，即对采样信号进行取模运算，随后对其进行门限判决即可。解调后，便得到对应包络的基带信号图，如图3-3所示，分别为停车杆落下和升起的序列。



（a）停车杆落下的序列

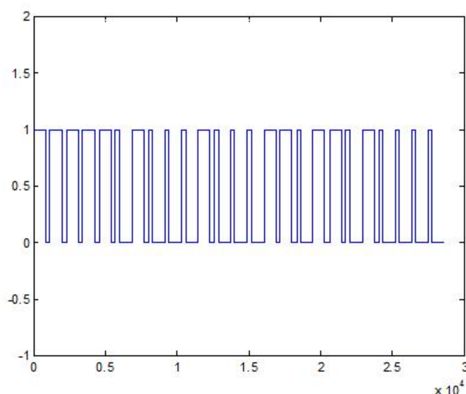


图3-3　非相干解调波形图

通过上面波形图可以看出，该脉冲调制具体采用的是脉宽调制，即PWM（Pulse Width Modulation），由两种占空比不同的矩形脉冲构成，其占空比与信号的瞬时采样值成比例。假设图3-3中宽脉冲用来表示符号“1”，窄脉冲用来表示符号“0”，由此可推断出停车杆落下的序列为：11111010001000110110100000，

停车杆升起的序列为：11111010001000110110010000。

根据以上信号的上升沿的样点序号，可以计算出每个符号所占用的采样点数，除以采样率8MS/s，就可以得到信号每个符号的传输周期为1205μs，即信号波特率为829.88Samples/s。

通过计算，可以得出宽脉冲维持高电平的采样数为891Samples，即891/1205＝0.739，占空比约为3/4。

窄脉冲维持高电平的采样数为285Samples，即285/1205＝0.237，占空比约为1/4。

下面我们用4位二进制数来表示一个符号位，也就是用1110表示符号“1”（占空比为3/4），1000表示符号“0”（占空比为1/4）。即对应的十六进制E表示宽脉冲，十六进制8表示窄脉冲。

每个符号展开可以表示为：

停车杆落下的信号为11111010001000110110100000，对应的十六进制比特向量为EEEE E8E888E888EE8EE8E88880。

停车杆升起的信号为11111010001000110110010000，对应的十六进制比特向量为EEEE E8E888E888EE8EE88E8880。

3.2 遥控信号重放攻击

因为遥控钥匙对停车杆每次发射的抬杆和落杆的信号是固定不变的，通过上一节对信号的分析，已经得到了这两个信号的比特信息。下面我们们就把破解出来的信号内容使用GNU Radio复现，产生停车杆升起和落下的信号，以控制停车杆。

编写PWM调制模块，使用GRC做出如图3-4所示的流图。

在图3-4中：

- “Vector Source”作为标准的向量输出源模块，可以输出自定义矢量，如停车杆落下的序列。参量Repeat决定是否重复地产生整组矢量数据。

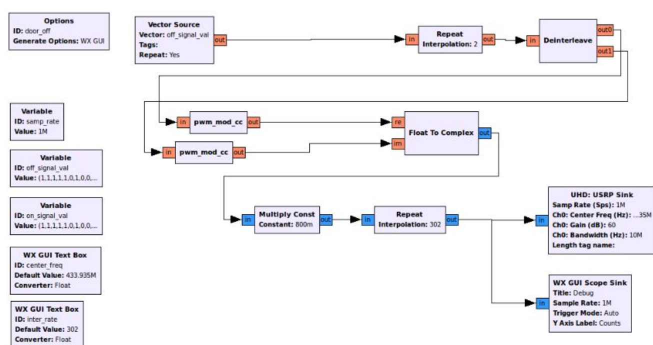


图3-4　PWM调制流图

- “Repeat”模块是将输入的矢量进行插值复制，即将＜a，b，c，...＞复制为＜a，a，b，b，c，c，...＞。

- “Deinterleave”解交织模块的作用是将“IQIQIQ...”数据分成“III...”与“QQQ...”两路数据，分别作为信号的实部与虚部传输给下一个模块。

- “pwm_mod_cc”为自定义的PWM调制模块，其作用是将符号“1”调制为十六进制F、符号“0”调制为十六进制“8”发送。

- “Float To Complex”模块的作用是将信号的实部与虚部合并为一路复数信号输出给下一级。

- “Multiply Const”模块的作用是将信号乘以一个常量，用来调整信号的幅度。

- “USRP Sink”模块是作为发射机的数据信宿。

- “WX GUI Scope Sink”模块用于查看发射出去的信号。

需要说明的是，发射机的采样率设为1MS/s，取决于在发射端之前增加了“Repeat”模块，每个bit重复302次，即采样率（1MS/s）＝比特率×302，比特率＝波特率（829.88Samples/s）×单个调制状态对应的二进制位数（4）。

完成上面的操作步骤后，使用USRP B210发射所复现的信号。为了验证结果正确与否，在另外一台PC上通过电视棒（RTL-SDR）来分别接收由B210和遥控钥匙发射的信号，进而通过HDSDR软件来观察信号的频谱，采集AM解调的信号图。

如图3-5和图3-6所示分别为遥控钥匙发射落下停车杆信号与B210发射复现信号的频谱图，可以明显地看出，由于遥控按键时间较短，所以在频谱图上体现为在中心频率附近产生一个亮点；而通过B210所发射的信号是连续、重复的，所以在其频谱图上体现为一连串的多个亮点。

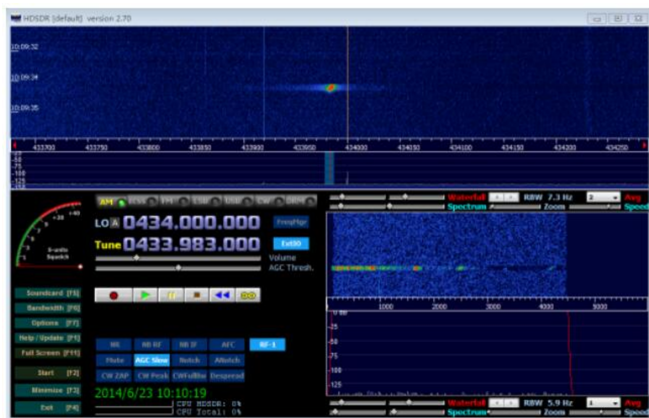


图3-5　遥控钥匙发射信号的频谱图

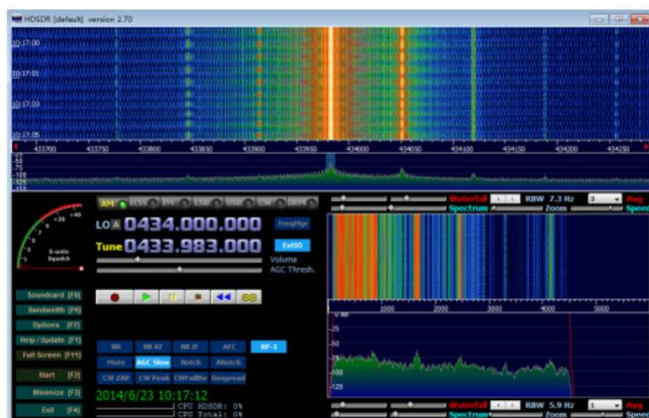


图3-6　B210发射信号的频谱图

将电视棒采集的AM解调信号保存为wav文件，使用Audacity软件分析两个波形是否一致。如图3-7所示，可以明显地看出遥控钥匙发射的信号与B210发射的信号波形一致。最后，便是在停车杆接收信号范围内通过B210发射复现的信号，结果是可以成功控制停车杆的落下与升起。



图3-7　停车杆落下信号的两个波形对比

发射信号的工具，除了USRP、HackRF以外，还有RfCat、Chronos手表。其中，后两种工具里有315MHz和433MHz频段的专用芯片。

如图3-8所示，就是使用Chronos手表控制停车杆的情形（http://v.youku.com/v_show/id_XNzMyMTI2NzE2.html？from＝y1.7-2amp;qq-pf-to＝pcqq.c2c）。



图3-8　使用Chronos手表控制停车杆

3.3 车库门固定码暴力破解

上一节介绍的是重放攻击方法，先录制信号分析，然后再发射同样的信号去控制接收端。本节将介绍在不能提前获得信号的情况下，如何暴力破解。

在我们身边经常使用无线遥控钥匙的还有车库门，而生活中常见车库门的遥控信号大多采用固定码技术，因此通过遍历的方式就可以暴力破解。下面我们来看Samy Kamkar在没有录制信号的情况下，直接暴力破解一个车库门需要多长时间（详见http://samy.pl/opensesame/）。

3.3.1　暴力破解的复杂度分析

固定码技术的缺陷就在于其密钥空间是极其有限的。如图3-9所示是车库门遥控钥匙的内部结构，左下方总共有12个拨码开关，一个12位密钥的遥控开关打开一个固定码字的车库门总共需要4096种组合。在这里，我们以常见的8~12位密钥为例。



图3-9　遥控开关的内部结构

我们观察到，通过点击一次按键，相同的控制信号会被连续发送5次，并且每个bit的发射时间为2ms，两个bit之间的时间间隔为2ms，所以发射一个12bit的控制信号需要12bit×2ms transmit×2ms wait×5times＝240ms。因此，暴力破解8~12bit的密钥组合总共需要发射的bit数为：

$$(((2^{12})\times 12)+((2^{11})\times 11)+((2^{10})\times 10)+((2^{9})\times 9)+((2^{8})\times 8))=88576\text{bit}$$

需要的时间为：

$$88576\text{bit}\times 4\text{ms}\times 5\text{transmit}=1771.52\text{s}=29\text{min}$$

由此可见，打开一个8~12bit长度密钥的车库门总共需要29分钟就可以完成（前提是知道该遥控钥匙采用的频率和带宽，当然通用的只有几种）。29分钟虽然不长，但是其实还可以将时间缩得更短。

同样的控制信号传输5次，是为了降低信号传输的干扰，目的是保证接收机可以接收到完整的控制信号。假设在信号不受到干扰的情况下，每个控制信号只需传输一次，如此一来，破解一个车库门的时间就会降低为原来的1/5！

$$1771.52s/5 = 354.304s \approx 6min$$

在实际测试中，Samy Kamkar发现去除码字之间2ms的等待时间，即对每个bit进行无缝传输，也可以达到目的。这样，整个传输时间又降低了1倍。

现在破解一个车库门的时间已经降到了3min。

$$1771.52s/5/2 = 177.152s \approx 3min$$

在实际数字电路中，车库门的接收装置中用来验证控制字正确与否是通过一系列的触发器级联的串行移位寄存器来实现的。移位寄存器的特点是在同一个时钟触发下，数据可以依次向左或向右移动。

令人兴奋的是，移位寄存器属于时序逻辑电路，即在任意时刻的输出不仅取决于当前的输入信号，还与之前的输入信号有关，因此可以保证数据不会在一个时钟周期下被完全清除。例如，假设打开某一车库门遥控需要发送的控制码字为“111111000000”。这时，我们发送一个13位的控制字“0111111000000”给这个无线接收装置，该装置首先检测“011111100000”（错误）；随后，由于系统采用串行移位寄存器，即寄存器一个时钟周期只能移动1位，下一个时钟周期检测到“111111000000”（正确）。因此，发送13个bit就可以检测两种不同的12bit密钥，而不是发送24个bit。同理，用12bit的测试码字可以同时去破解8~12bit长度的密钥。由此可见，发送密钥组合的bit总数即测试码长度还可以继续降低。

在这里，我们以3bit长度的测试码来进行说明，具体缩减测试码长度的方法如下：

暴力破解一个3bit密钥需要测试的组合形式：

1bit 10bit 20bit 30bit 40bit

bit：01234567890123456789012345678901234567890123456789

000---001---010---101---011---111---110---100--- 总共48bit时间

除去中间等待的bit时间，就变成下面这样：

bit：01234567890123456789012345678901234567890123456789

000001010101011111110100　　　　　　总共24bit

进行bit压缩：

000 011
001 111
010 110
101 100
0001011100　　　　　　总共10bit

不难看出，这10个bit中包含了3bit密钥的任意组合。细心的读者可以发现，在进行bit压缩时，000~111并没有采用顺序的方式进行组合排布，而是采用一种巧妙的方式排序，源于在此引用了荷兰数学家Nicolaas Govert de Bruijn的算法，该种排序方法称作De Bruijn序列。De Bruijn序列的特点是可以保证遍历n阶移位寄存器的所有可能状态（汉密顿路径）。

因此在使用该算法的情况下，破解一个8~12bit密钥的时间仅为：

$$(2^{12}+11)\times 4ms/2=8214ms=8.214s$$

3.3.2　固定码暴力破解的硬件实现

理论分析完之后，下面来看看硬件是如何实现的。

Samy Kamkar采用的工具是一款叫作IM-ME的玩具，这款玩具主要用于短距离间通信。之所以选用这款玩具，是由于其采用TI的CC1110芯片，并且设有LCD显示屏、键盘、背光、电池等便利因素，如图3-10所示。而且IM-ME可以称作是口袋频谱分析仪，该设备支持的频率范围为281～361MHz、378～481MHz和749～962MHz。这就涵盖了生活中大多数活跃的频率范围，其中在美国主要包括ISM、LMR、电视、对讲机、手机和业余的频段。

除此之外，IM-ME还支持GoodFET，这样就可以通过开源的JTAG对设备进行调试。如此一来，更有利于破解工作的硬件环境。图3-11所示，即在IM-ME设备上焊接一些测试导线，以方便与GoodFET设备相连。连接后，便可通过GoodFET对IM-ME进行软件编程调试。其主要用于将程序下载至TI的CC1110芯片上。

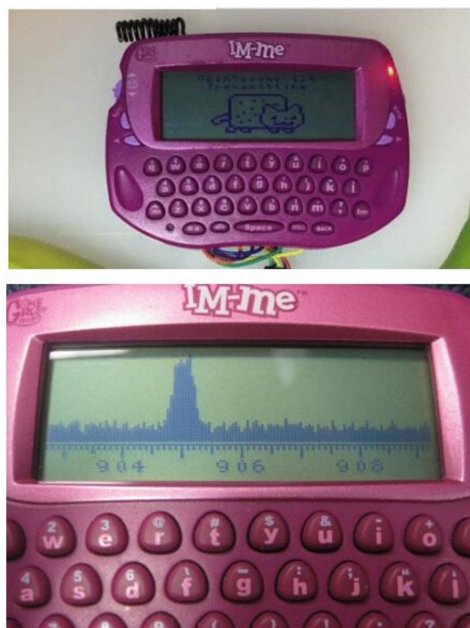


图3-10　破解工具IM-ME

Samy Kamkar给这套系统命名为OpenSesame。其中，OpenSesame软件部分的代码可以参考Samy Kamkar的github网站：https://github.com/samyk/opensesame。

关于频率的设定。通过维基百科可以查到目前大多数这种无线装置采用的载频为300～400MHz，然而，这是否意味着我们要在100MHz频率范围内漫无目的地发送同样的信号进行实验呢？实际情况证明并不需要这样复杂，通过翻阅FCC关于固定频率发射机的文档可以知道只有少数的频率正在使用，主要是300MHz、310MHz、315MHz、318MHz和390MHz，所以只需要在这些频点上尝试暴力破解就可以了。

通过实测还发现，其实大多数接收器都缺少一个带通滤波器，因此可以允许更大范围带宽的信号通过，通常在中心频率左右，带宽2MHz的信号均可识别。



图3-11　连接GoodFET

此外，由于大多数车库门的遥控发射信号均通过ASK/OOK产生，故接收器的解码装置也一致。因此，同一个测试序列可以同时破解多个车库门！当然，这些车库门的无线接收器需要满足信号传输距离和密钥长度的要求。

3.4　汽车遥控钥匙信号安全分析

汽车钥匙的无线功能给我们带来了便利，我们再也不用掏出钥匙插到钥匙孔来打开车门了。无线技术发展到现在，有一些车钥匙的无线功能甚至可以做到当你走到主驾驶门时，车门自动解锁；当你带着钥匙离开时，汽车会自动上锁。这些都是通过各种形式的无线信号实现的，那么汽车无线钥匙的“信号”长什么样呢？

常见的汽车无线钥匙一般工作在两个频点，即315MHz和433MHz。调制方式比较常见的有两种，即ASK和FSK。复杂一点的有双频点FSK，还有多频点ASK。

在这里，我们使用RTLSDR硬件和HDSDR软件，把频点设在汽车钥匙使用的频点上，按动汽车钥匙的按键，就可以看到像下面这样的频谱图。

如图3-12所示是一辆奔驰车钥匙的信号频谱图。从图中可以看出，信号的频段是433MHz，中心频点是433.96MHz。信号在两个频点有很强的能量，因此这是双频点FSK信号，即用载波频率的变化来表征被传信息的状态，被调载波的频率随二进制序列0、1状态而变化。中心频点大约为433.97MHz。这个车钥匙每次按键无论时间长短，只产生两段信号，按下去时产生一段，松开时产生一段。

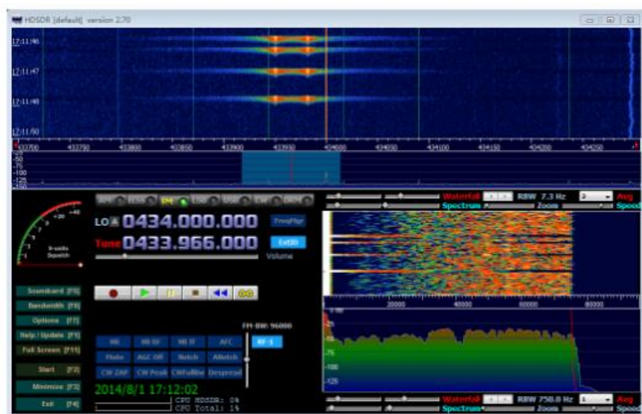


图3-12　某奔驰车钥匙信号频谱图

如图3-13所示是一辆奥迪车钥匙的信号频谱图。从图中可以看出，信号的频段是315MHz，中心频点是315.04MHz。将HDSDR设为AM解调，按一次录音按键，录一段解调之后的波形，保存为wav文件。

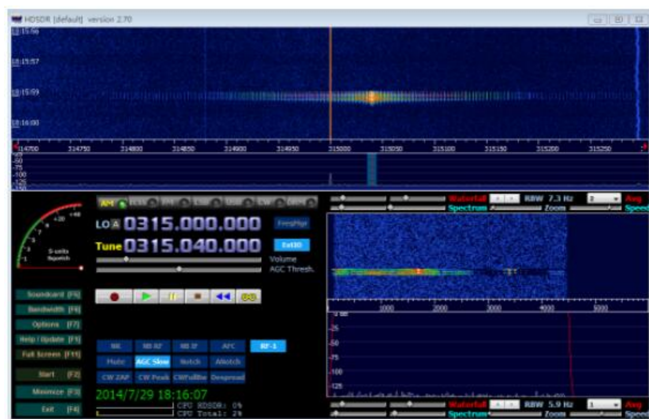


图3-13　某奥迪车钥匙信号频谱图

接下来，采用音频分析软件Audacity打开波形文件（或者使用其他的编辑wav文件的软件），就可以看到如图3-14所示的波形，这就是一次按键发出的信号，会发射两段信号。

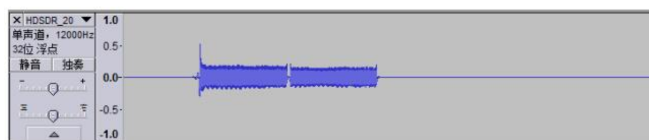


图3-14　按键的信号波形图

如果按下的时间长一点，就会如图3-15所示，出现了多段信号。

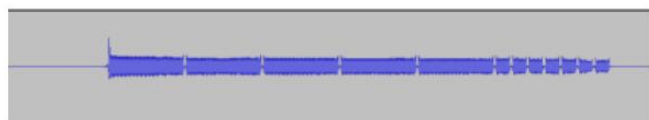


图3-15　长时间按键的信号波形图

在按下的时间内，钥匙持续发出一段段信号。这里面的每一段信号都是一样的，重复发送。下面我们将信号放大展开，仔细看一下每一段信号的样子。如图3-16所示，前面部分是一些重复的脉冲，作用

和前导码差不多，就叫它同步引导序列吧，用来提示接收器有信号即将到来，还可以获得时钟信息；后面部分是有效的数据。

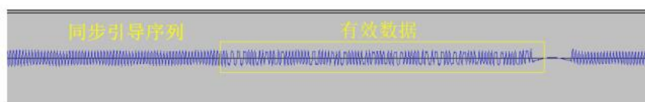


图3-16　信号数据包波形图

下面我们把一次按键的两段信号拖到一起，比较一下，如图3-17所示。

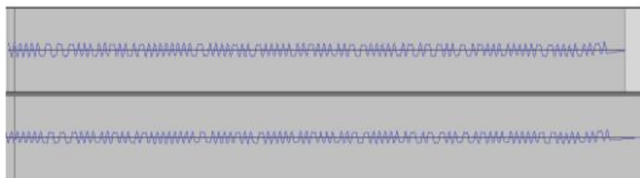


图3-17　同一次按键的两个数据包波形对比

可以看到，一次按键发射的两个数据包的信号是完全一样的。

我们再把两次按键的信号放在一起进行比较，结果如图3-18所示。



图3-18　两次按键的数据包波形对比

可以看到，两个码前面部分是一样的，尾部是一样的，中间不一样。由此可知，两个信号很明显是不一样的。也就是说，汽车钥匙每次按键，发射的信号都是不一样的。但这并不是说所有的车钥匙信号波形都是这样的格式，比如奔驰车钥匙信号的特点是，前一部分的码是固定的，后一部分的码是变化的，并没有固定的尾部。这些变化的码字就是大名鼎鼎的“滚动码”（Rolling code）。

滚动码的原理大致是这样的：

- 滚动码是一个周期很长的伪随机码。例如一个40bit长度的滚动码就有240种码字组合。而现在大部分车钥匙的码长都大于40bit。

- 车钥匙里存有当前的滚动码。当车钥匙按下时，滚动码加上功能码（比如是开锁、解锁，还是开后备箱）一起发送给汽车。

- 汽车里也存有当前的滚动码。当它接收到同样的滚动码时，就执行相应的开锁之类的操作；如果收到的码不匹配，就不做任何动作。

- 车钥匙和汽车里的滚动码是保持同步的。

- 当车钥匙距离车很远时，有人不小心按了几次车钥匙，车钥匙里的随机码就会前进好几步。此时跟车内的码就不同步了。为了解决这个问题，汽车允许接收当前码之后的（比如）几百个码。只要车钥匙发送的码在这个窗口之内，汽车都认为是有效的。

- 如果车钥匙被误按超过设定的几百次，那么车钥匙和车里的滚动码就彻底失去同步了。这时，就需要查阅汽车的使用手册，找到恢复同步的方法。

从上面的原理可以看出来，如果能得到汽车当前滚动码“之后”的一个码，只要在窗口之内，就可以把车打开。那么怎样才能得到一个有效的、未被使用的码呢？比较简单的方法是像图3-19所示这样的。



图3-19　隔离录制重放攻击

录制一段信号，然后到汽车那边去播放。

或者是像图3-20所示这样的，一边录，一边播……



图3-20　实时录制重放攻击

不管怎样都需要接触到车钥匙，所以打开车门也并不是件容易的事情。至于录制工具有很多种，用前面讲到的电视棒就可以。播放工具也有很多种，比如RfCat，如图3-21所示（http://int3.cc/products/rfcat）。

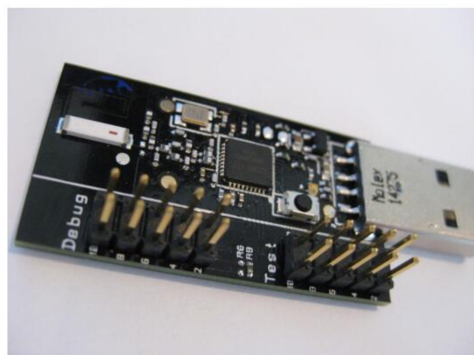


图3-21　播放工具RfCat

再比如利用TI的C1111芯片，Unicorn Team做的一个RF收发分析器，如图3-22所示（增加了PA和SMA）。

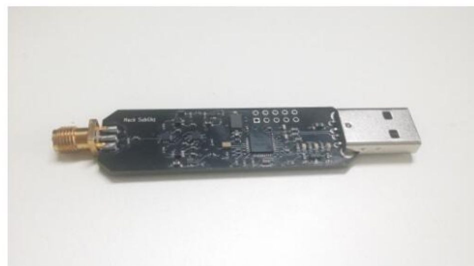


图3-22　Unicorn Team RF收发分析器

当然，还有在3.2节提到的破解固定码所用的Chronos手表，如图3-23所示。如果用手表发射信号，还需要对接收到的信号通过Matlab进行解调，得到信号的脉宽并将信号以十六进制编码来重新配置手表。首先将录制的wav文件通过Matlab算法对信号解调，可以得到信号的符号周期、脉宽、波特率及信息总长等信息，用于手表的参数配置；然后就是将量化后的二进制编码转换成十六进制编码；最后通过一个叫作“USB access point”的硬件与手表进行通信。



图3-23　Chronos手表

前面提到了信号解调，我们在3.1节中简单地介绍了2ASK信号的非相干解调方法，有兴趣的读者可以尝试从截获的wav文件解码来分析。大概需要经过以下9步：

（1）滤去信号的直流成分。

（2）把复信号转化为功率值，abs(x)^2。

（3）做低通滤波，把功率信号中的高频噪声去掉。

（4）功率归一化，为了方便下一步做门限判决。

（5）门限判决，以0.5为门限，高于0.5则判为1；否则为0。

（6）根据上升沿、下降沿，确定出符号周期。

（7）根据符号周期对判决结果取样。

（8）搜索同步引导序列，确定数据段的起始位置。

（9）最后经过曼彻斯特进行解码。

需要注意的是，采用Chronos手表发射信号时只需要解调即可，不需要再经过曼彻斯特解码，因为我们只需要再生这个信号，照着原信号的波形发射出去就可以了。

2FSK信号则是用二进制数字信号来控制发送不同频率的载波。即传“1”信号时，发送频率为f1的载波；传“0”信号时，发送频率为f2的载波。可以理解为2FSK信号是两个不同载频交替发送的两个2ASK信号之和，因此2FSK信号的解调相当于带通滤波器再加上ASK的包络检波。然而，并不是所有的钥匙都只有这两种编码方式，一些复杂的调制方式还需要通过FFT进行频谱分析。

那么有没有不需要接触钥匙，就能得到一个有效码的方法呢？

当然也是有的，那就是一些高级方法了。接下来就进入密码学的学术领域。

http://www.cosic.esat.kuleuven.be/keeloq/

这个链接的内容是2007年以色列和比利时的几个研究者，找到的一种破解滚动码的方法。该方法需要先花大约1小时的时间对钥匙进行65536次试探，解出64bit中的36个bit，然后再花几秒钟就可以完全破解钥匙的滚动码。完全破解的意思就是，这个伪随机码的全部规律都已经被破解了，可以知道下一个码是什么。

在实践中，需要用相应的芯片（也就是汽车端配对的芯片）伪造一个transponder and encoder，让它工作在IFF（Identify Friend or Foe）模式，对车钥匙发出试探信号（challenge），车钥匙发回一个response。车钥匙每产生一次response需要60ms或90ms，因此收集65536个challenge/response消息对，需要65min或者98min的时间。那么，车钥匙能够支持IFF模式的距离是多远呢？应该是非常近的距离。所以场景差不多是图3-24所示这样的。



图3-24　破解滚动码方式

2008年，德国鲁尔大学的几个研究者使用side-channel attack（中文有翻译为旁路攻击，也有叫边信道攻击）的方法，通过分析解码控制器的功耗解出了KeeLoq密码（http://www.emsec.rub.de/keeloq）。因为控制器在做不同运算时消耗的功率是不一样的，通过测量控制器的功耗，结合KeeLoq的算法（这种算法基本上是很成熟和固定的），可以推导出控制器中存储的密钥是什么。他们的研究成果，把破解KeeLoq密码的时间缩短到了几秒钟。



图3-25　KeeLog算法破解

看起来这种方法是最牛、最高效的。但是仍然要接触到钥匙，还要懂得怎么测量控制器的功耗，分析出什么时候算乘法、什么时候算加法，目前全世界懂这个技术的人也不是太多。

所以，开车门并不是那么容易的。现实中，小偷的开车门方法其实是图3-26所示这样的。

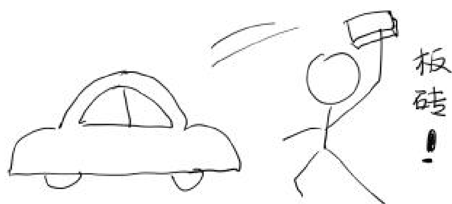


图3-26　小偷开车门方法

以上说的都是通过钥匙的无线信号解锁的方法。现在的车越来越智能，出现了用蓝牙连接、用手机APP通过蜂窝网络连接等各种新的开车门方法，这些不在本文讨论的范畴。但是往往一个系统越复杂，暴露的攻击面越多，可能出现的漏洞就越多。

另外，开关车门和发动汽车是两码事，分别由两个系统管理着。所以即使打开了门，也很难把车开走。你以为扯出两根电线短路一下就能发动汽车吗？那是电视剧！不过损失车内财物就在所难免了。

3.5 汽车胎压传感器系统安全分析

近几年汽车电子的飞速发展引起了人们的广泛关注，越来越多的ECU控制单元被应用在汽车上，从而实现全车的智能化控制，整车的安全性愈加至关重要。而胎压正常与否对于行车安全有着至关重要的作用，2002年，由于凡世通（Firestone）轮胎的质量问题，造成了超过100人死亡和400人受伤的事件，引起了汽车界和美国政府的高度重视，凡世通公司被迫收回650万只轮胎。据公安部统计，在中国高速公路上发生的交通事故有70%是由于爆胎引起的，而在美国这一比例则高达80%。从停车杆的无线信号讲到汽车遥控钥匙，现在我们将目光放到汽车胎压这一严重威胁安全的问题上。

如何防止爆胎已成为安全驾驶的一个重要课题。？

胎压过高和胎压过低都会对行车安全造成危害，胎压过高会使轮胎太硬，其减震效果变差，乘坐舒适性变差；而胎压过低，则是车胎漏气的前兆。由于轮胎压力变化通常是一个渐变的过程，即使异物刺破轮胎而导致的轮胎泄气也是一个持续的过程，因此通过实时监测轮胎压力，并在轮胎压力出现异常后的第一时间报警，能够为驾驶员正确处理突发情况争取宝贵的时间，从而保证行车的安全。

对于胎压监测，有些车已经自带了胎压监测系统，称作TPMS（Tire Pressure Monitoring System），如果出现问题则会自动报警。胎压监测系统需要在车胎上装有带讯号发射功能的传感器，如图3-27所示，可以提供实际的胎压数值，有些还可以提供车胎的温度。除了传感器以外，还需要有讯号接收器，该接收器安装在车上，两者以无线电连接，用来向驾驶员提供目前胎压是否正常的情况。



图3-27　胎压传感器

在这里我们研究的问题就是，传感器与接收器之间的无线通信是否安全呢？

我们购买了一款外置的、可由车主自己安装的胎压监测系统。为了方便在实验室内研究胎压传感器的信号，我们把胎压传感器绑在自行车轮子上，让轮子转到一定的速度，通过电视棒就可以观察到胎压传感器发射的信号。

图3-28显示的是捕捉到的传感器发射的信号。

通过对传输信号的分析，可以看到信号的中心频点大约是433.969MHz，是一种2FSK信号。频谱中心左边的第一个主瓣的位置在433.9598MHz，右边第一个主瓣的位置在433.9781MHz；左边第二个主瓣的位置在433.9493MHz，右边第二个主瓣的位置在433.9873MHz。因此这个FSK信号的频率偏差（frequency deviation）大约是10kHz。

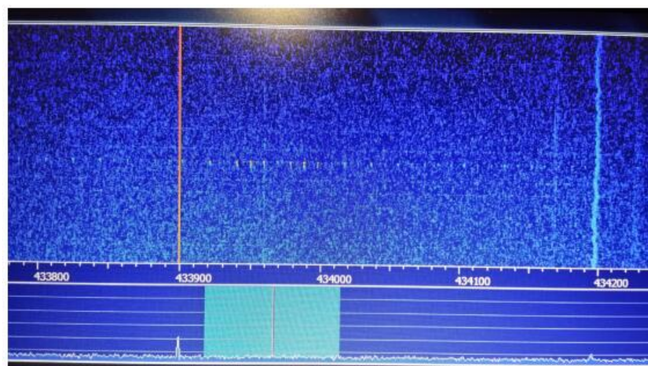


图3-28　传感器发射的信号

接下来，录下FM解调的信号保存为wav文件，通过Audacity观察，可以看出信号主要分为4段，后两段是前两段的重复，如图3-29所示。



图3-29　胎压信号时序波形图

进一步用Matlab解调数据，2FSK解调的方式为：

（1）对信号做FFT变换，得到对应的频谱图，找出频谱中能量最大的两个波峰，这就是2FSK中用来表示“0”和“1”的两个载频f_1，f_2。

（2）用低通滤波器滤掉高频，或者用高通滤波器滤掉低频。

（3）然后按照2ASK解调的步骤解调，请参考3.4节中2ASK的非相干解调方法。

最后经过曼彻斯特解码，通过分析可以得到数据比特。其数据的符号速率是19.2KS/s，比特速率是9.6Kbps。因此，发送一次数据包（包括第2次重发）的总时长约为20.83ms。

胎压数据包格式如图3-30所示。

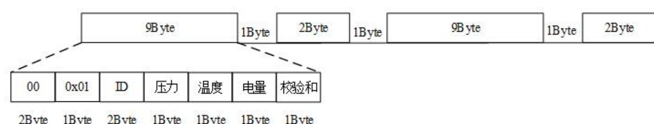


图3-30　胎压数据包格式

最前面有若干全0的前导信号。然后是9个字节的数据包，接下来停1个字节，发2个字节，再停1个字节，重复发一遍。我们主要关心这9个字节中后面7个字节的内容。

下面列举出捕捉到的一些数据中有效的7个字节内容：

1 223 59 56 63 106 252

1 223 59 56 64 106 253

1 223 59 56 64 107 254

1 223 59 56 63 107 253

1 223 59 56 63 121 11

1 223 59 57 64 106 254

1 223 59 56 63 107 253

1 223 59 56 63 107 253

这是采集的样本数据的一部分，其中每个十进制数字占用一个字节大小。值得注意的是，比特流发送的顺序是从低位（LSB）到高位（MSB），即每个字节的最低比特先发送。本次采样内容主要为行驶中得到的正常胎压的样本。

从以上样本可知，第2、3字节是轮胎传感器的ID。经过分析发现，最后一个校验字节等于前面6个字节以单字节求和的结果，校验字节取低8位。

例如，最后一个样本的校验值是：

1 + 223 + 59 + 56 + 63 + 107 = 509 = 111111101 => [11111101] = 253

第五个样本的校验值是：

1 + 223 + 59 + 56 + 63 + 121 = 523 = 1000001011 => [00001011] = 11

因此，在了解了胎压传感器的数据格式之后，就可以伪造任意的数据包了。当然，前提是轮胎传感器的ID号必须正确，否则接收模块是根本不会处理的。

首先我们看图3-31，该款胎压监测报警器在不同状况下的显示情况。



图3-31　报警器在不同状况下的显示情况

然后，就可以通过Matlab按照以上数据包格式产生异常的信号文件。接下来，继续使用无线电设备USRP B210将数据发射出去，采样率设置为19.2KS/s。

由于只攻击了一枚传感器，从图3-32可以看出左前方的胎压数值不正常。我们分别制造了一些异常情况，即胎压过高、胎压过低、轮胎温度过高和传感器电量不足，主要是通过对第4、5、6字节尝试不同的数值，第4、5、6字节分别表示轮胎压力、轮胎温度、电池电量。



图3-32　胎压逆向攻击

现在我们掌握了胎压信号攻击的方法，试想能否在路边不断发射0胎压信号，使经过的车辆产生报警呢？因为每个胎压传感器都带有一个ID号，这个ID号对我们来说是未知的。接下来，我们分析一下穷举爆破的耗时。

通过胎压的数据包格式可以知道轮胎传感器的ID占用2个字节，即16比特，因此有2^{16}种可能。而每发送一次数据包，加上前面的前导码部分，总长度大约为21ms，假设每两次发送之间暂停21ms，那么尝试所有可能的ID的总时长为：

$$2^{16} \times (21 \times 2)/1000/60 = 46\text{min}$$

这个破解时间太长了，等你破解完，车已经开出百里之外了。即使把每次发送数据包的时间缩短到11ms，也就是每个数据包只发送一次，13个字节，要消耗的总时长也将达到：

$$2^{16} \times (11)/1000/3600 = 12\text{min}$$

所以，想要通过穷举法，给一个行驶经过的车发送错误的胎压信号，是不现实的。那么还有其他高招吗？答案肯定是有的，就是监听加伪造重放的办法。

首先让一个程序不停地侦听胎压信号，一旦收到一个数据包，就可以立刻解码得到轮胎传感器的ID值，然后通过发射程序自动修改其中的胎压值、温度值或电池电量值某一项，马上回放出去，自然就会使得TPMS系统发出警报。当然，仅仅是警报而已，不会威胁到行驶车辆的安全。

总的来说，目前市面上采用的TPMS系统主要是明文传输，且校验方法极其简单，很容易被窃听并伪造重放。可能因为胎压数据不属于隐私程度很高的数据，伪造数据也不会对车辆安全产生严重威胁，因此，设计人员才使用这么低的安全设防！设想，假如不法分子在路边利用这种方式让车主产生错觉，随后当车主下车查看车况时趁机打劫，不得不说这也会严重威胁到车主的人身、财产安全。因此，与人们生活息息相关的安全性都不容小觑。当然，你也可以脑补这样一部电影大片，一个通晓无线技术的工程师被歹徒绑架到某辆车上，当歹徒胁迫这个工程师用他手中的无线设备进行恐怖袭击时，这个工程师就可以通过攻击这辆车的TPMS系统使驾驶员等分心，趁歹徒下车查看车况时，溜之大吉。

最后，从设计原则的角度来看，建议：

- TPMS系统一定要与车辆的中控系统有足够的隔离度。也就是说，TPMS系统发出警报，就足够了。不能对中控系统“输入”进一步的指令，产生一些危及车辆安全的其他行为。

- TPMS系统与中控系统连接时要注意内存管理的问题，防止大量伪造数据包输入，造成中控系统故障。

- 在综合考虑产品成本和易用性的前提下，适当提高TPMS信号的加密等级。

第4章　航空无线电导航

在讲航空无线电导航的安全性问题之前，先简单谈一谈与航空有关的无线电技术。航空飞行涉及的无线电设备种类繁多，使用的频谱资源广泛，可以简单地分为两类：通信设备和导航设备。通信设备主要是用于飞机与飞机之间、飞机与地面之间的通信；导航设备包括卫星导航、地面信标辅助的导航等。

民用航空无线电频率的电磁干扰很容易影响飞行安全，带来的隐患不容忽视。很多航空通信设备使用的是很原始的AM、FM调制等方法，抗干扰能力很差。经常有媒体报道各种奇特的干扰航空频率的事件，例如工地内一台塔吊监视器的视频无线传输设备干扰了附近机场的雷达站；大功率无绳电话干扰了调幅航空电台；机场候机大厅的霓虹灯字，因霓虹灯使用时间过久，电子元件老化导致屏蔽能力变差，电磁辐射干扰了机场的塔台。

那么，已经到了21世纪，为什么航空无线电设备仍然在使用抗干扰能力差的技术呢？其中一个原因是无线电设备需要兼容各种飞机，老机型也必须支持；另一个原因是在极低信噪比的条件下，AM调制传送的模拟语音信号反而是可靠性很高的一种通信手段。因为人耳能够在很模糊的背景噪声中辨识出人的呼叫声，这里依靠的是人耳和脑的强大的“解码”能力。

航空无线电还有一个特点就是“不加密”。航空电台的呼叫是不加密的，如果你愿意，可以轻松地收听不加密的空地VHF语音通信内容。飞机发射的报告自己位置和状态的一些信号，也都是不加密的，目的就是尽可能让更多的人听到“我在这里，我还好”。所以，我们也可以轻松地用一些小设备，听到附近飞机的报告。

4.1 ADS-B系统简介

本章我们讲解一个例子：伪造一个基于1090ES的ADS-B广播。

4.1.1　ADS-B是什么

我们先看看目前常用的飞机位置监控系统主要有哪些，如表4-1所示。

表4-1 目前常用的飞机位置监控系统

类型	是否独立工作	是否需要相互配合工作
一次监视雷达（PSR）	独立：由雷达独立监测	不需要飞行设备提供雷达回波
二次监视雷达（SSR）	不独立：由飞机提供应答信息进行监测	需要飞行设备工作在 ATCRBS 的应答状态
自动相关监视广播（ADS-B）	不独立：　由飞机提供监控数据	需要飞行设备具有 ADS-B 功能

PSR（Primary Surveillance Radar）即“一次监视雷达”，该设备的特点是不需要飞机的应答，可以直接监视飞机位置。

SSR（Secondary Surveillance Radar）即“二次监视雷达”，该设备的特点是需要飞机的应答，由飞机自己提供其位置信息。

ADS-B（Automatic Dependent Surveillance-Broadcast）即自动相关监视广播，顾名思义，该系统无须人工操作或询问，可以自动地（每秒1次）从机载设备获取相关参数向其他飞机或地面站广播飞机信息，内容主要涉及位置、高度、速度、航向、识别号等，以供管制员对飞机状态进行监控。它衍生于ADS（自动相关监视），最初是为越洋飞行的航空器在无法进行雷达监视的情况下，希望利用卫星实施监视所提出的解决方案。

可供ADS-B系统传输数据使用的数据链有4种，其中Mode S（S模式）数据链、VDL-4数据链和UAT数据链是大多数ADS-B系统所选择的。S模式是飞机的应答机的一种工作模式。应答机有若干种模式，如模式1，2，3/A，4，5，B，C，D和S模式，这些模式使用的都是脉冲调制信号。A模式和C模式最简单，也是目前使用最广泛的模式。S模式的特点是给每架飞机都设置了一个24bit的地址码，可以实现点名式的询问和应答。

S模式的应答机配合二次监视雷达工作时，地面雷达对飞机发射询问信号，频率为1030MHz；飞机对地面发射应答信号，频率为1090MHz。

4.1.2　1090ES的含义

前面提到，我们要伪造一个基于1090ES的ADS-B广播，那么1090？ES又是什么？

1090ES（1090MHz？Extended？Squitter）是基于S模式应答机上的一种集中报文格式，它的最大优点是原有的？S模式应答机可升级为支持ADS-B系统的载体，且可以提供1Mb/s的带宽。原本S模式只发送飞机的高度和编号信息，后来为了让飞机广播更多的信息，就扩展了消息的长度，可以广播飞机的位置、速度、高度、呼号等信息。

因此，ADS-B Mode S1090ES的意思是：使用ADS-B协议、物理层是S模式、Extended Squitter、工作在1090MHz频率。

开发出ADS-B的目的主要是：一次监视雷达和二次监视雷达覆盖的区域有限，建设雷达站的成本比较高；而ADS-B成本低、建设速度快，可以快速覆盖大面积地区。1090ES数据链改进了传统的二次雷达传输方式，使得自己拥有较强的数据传输能力，同时也加入了独特的地址位信息来表明发送者的身份。S模式的ADS-B系统不仅可以用于飞行器与地面基站之间的信息传递，也可以用于飞行器之间的信息交互。

正因为ADS-B技术简单、成本低，所以才有大量的黑客和无线电爱好者自己制造ADS-B接收机。

需要强调的是，ADS-B只是监视飞机位置的众多系统中的一种。

4.2　ADS-B信号编码分析

ADS-B Mode S1090ES协议可以在参考文献[1]~[6] 中找到。完整的协议内容很多，这里只介绍一种最典型的报文，即飞机发射的空中位置的报文。它的载波频率是1090MHz。

4.2.1 调制方式

ADS-B信号将0、1脉冲直接调制在载波上。在1s的时间内，脉冲在后半部分表示0，在前半部分表示1，也可以用01表示0，10表示1。

一条完整的ADS-B消息长度是120s，总共有112bit数据位信息，由两部分组成：前导脉冲部分和数据脉冲部分。

前导脉冲位于开头的8s，其作用是为后面数据块的解码提供信号到达时间和脉冲参考电平等参数，解码器根据这些参数进行数据块的解码并且判断出数据位的“0”和“1”取值。

如果前导脉冲部分以0.5s脉冲表示，那么图4-1可以记为：

10 10 00 01 01 00 00 00

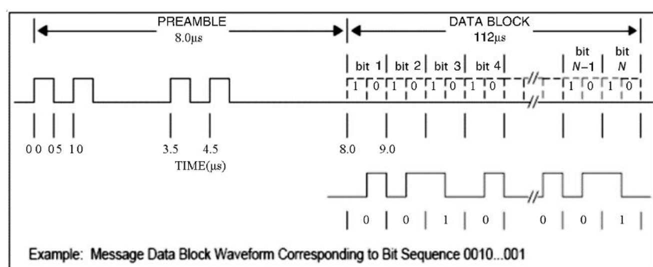


图4-1　前导脉冲部分

4.2.2　报文格式

数据脉冲位于前导脉冲之后，数据块为112bit报文，第一个脉冲的位置出现在8s位置。对于数据报文，参考文献[6] 给出了一个非常好的例子。如下所示，有两条ADS-B消息，每条112bit：

8D 75804B 58 0FF 2CF7E9BA6F701D0

8D 75804B 58 0FF 6B283EB7A157117

消息的各字段及其含义请见表4-2。

表4-2　消息的各字段及其含义

比特数	字段含义	数据含义
5bit [1:5]	DF（Downlink Format），下行数据链格式字段	本例中，DF = 10001 = 17，表示用于 S 模式应答机发射的 ADS-B 消息
3bit [6:8]	CA 能力字段（用于 DF=17 格式），表示应答机能力	本例中，CA = 101 = 5，表示至少有通信 A 与通信 B 能力，且飞机位置在空中
24bit [9:32]	ICAO（International Civil Aviation Organization），分配给飞机的一个 24bit 的身份信息	本例中，ICAO = 0x75804B，表示 CEB [5J] Cebu Pacific Air Registration RP-C3191 Airbus A319，菲律宾宿务太平洋航空的一架空客 319 飞机
5bit [33:37]	TC（Type Code），表示后面数据的类型	本例中，TC = 01011 = 11，表示这是一个空中位置消息

续表

比特数	字段含义	数据含义
3bit [38:40]	STC（Sub Type Code），表示后面数据的类型（与 TC 共同作用）	本例中，STC = 000，表示该子字段不存在，消息的类型通过 Type 子字段和 SubType 子字段共同作用
12bit [41:52]	Altitude，高度信息	本例中，Altitude = 000011111111，高度编码见 4.2.3 节
1bit [53]	T，Time 表示时间是否与 UTC 同步	本例中，T = 0，表示没有与 UTC 同步
1bit [54]	F，CPR 格式，表示后面的 CPR 位置消息是奇数秒还是偶数秒的。0 表示偶数秒，1 表示奇数秒	本例中，第 1 帧的 F = 0，第 2 帧的 F = 1
17bit [55:71]	Latitude，纬度信息	第 1 帧为 10110011110111111 第 2 帧为 10101100101000001 CPR 经纬度编码见 4.2.4 节
17bit [72:88]	Longitude，经度信息	第 1 帧为 01001101110100110 第 2 帧为 11110101101111010 CPR 经纬度编码见 4.2.4 节
24bit [89:112]	CRC，校验信息	CRC 校验见 4.2.5 节

4.2.3 高度编码

高度数据的编码方式在参考文献[10] 的2.2.13.1.2节有标准的描述。下面给出一个简单的解释。

高度编码的12bit：

| 1 | 2 | 3 | 4 | | 5 | 6 | 7 | 8 | | 9 | 10 | 11 | 12 |

Q bit

其中的第8比特称为Q比特，Q＝1时，表示这个高度数值以25英尺为单位；Q＝0时，表示这个高度数值以100英尺为单位。飞机高度低于50175英尺时，都使用25英尺为单位。

去掉Q比特之后，剩余的比特构成二进制数N。

飞机的高度为：N×25-1000（英尺）

对于本例而言，Altitude＝000011111111。Q＝1，因此以25英尺为单位。去掉Q比特，剩余的比特为00000111111＝127，因此飞机的高度为127×25-1000＝2175英尺。在Q＝1的情况下，仅能提供范围为-1000～＋50175英尺的代码值，高于50175英尺的高度编码采用Q＝0的情况，即以100英尺增量为单位。

4.2.4 CPR经纬度编码

在1090ES ADS-B系统中，为了提高ADS-B消息的传输效率，扩展的S模式断续震荡采用简洁位置报告（CPR，Compact Position Reporting）形式来有效编码经度和纬度消息，在每个消息中不再发送长时间不变的高位。例如，在纬度的二进制编码中，有一位用来指代飞机处在南半球或北半球，由于飞机位置不会发生突变，故该位在长时间内不会发生变化，因此不需要重复发送这一位。然而，假如不发射高位的话，那么地球上很多位置的编码会相同，而CPR编码就是通过两种略微不同的方式来实现的，分别为偶形式编码和奇形式编码。通过在短时间内接收到两种编码的消息可以确定飞机的准确位置。

如图4-2所示，为了确定飞机的具体位置，CPR算法对地球的表面进行位置划分，将地球的球体保持一个近似固定的距离分辨率。经度的取值范围为东经180°至西经180°，纬度的取值范围为南纬90°至北纬90°。经度和纬度各用17bit表示，且经纬度的划分也都采用了奇偶两种方式，其中纬度区段采用了定长的区段长度；经度区段从本初子午线向东环绕地球，其长度随着纬度的绝对值增大而减小，赤道附近经度区段的尺寸最小，两极经度区段的尺寸最大。

纬度区段划分公式：

$$D_{\text{lat}}(i)=\frac{360}{60-i}$$

其中，D_{lat} 代表纬度区段的长度，i表示奇偶参数。

经度区段划分公式：

$$D_{\text{lon}}(i)=\begin{cases}\dfrac{360}{\text{NL}(\text{lat})-i}, & \text{NL}(\text{lat})>i\\ 360, & \text{NL}(\text{lat})=i\end{cases}$$

其中，D_{lon} 表示经度区段长度，i表示奇偶参数，lat是纬度值，范围为[-90，90]。

一个纬度处的偶经度区段数量为：

$$NL(Rlat_i)=\begin{cases}59, & lat=0\\ 1, & |lat|>87\\ 2, & |lat|=87\\ \text{floor}\left(2\pi\left[\arccos\left(1-\frac{1-\cos\left(\frac{\pi}{2NZ}\right)}{\cos^2\left(\frac{\pi}{180}|Rlat_i|\right)}\right)\right]^{-1}\right), & \text{others}\end{cases}$$

其中，NL代表一个纬度处的偶经度区段的数量，一个连续的经度区段对应一个NL值，当NL值发生变化时，纬度被称作“NL Transition Latitude”，NL Transition Latitude的值和偶经度区段边界通常不会重合。

划分区段后，地球表面可量化的位置点数还取决于二进制编码的长度。在ADS-B的空中位置报文中，CPR编码的纬度和经度均采用17bit量化。因此，纬度区段和经度区段的像素点个数均为2^{17}＝131072个。

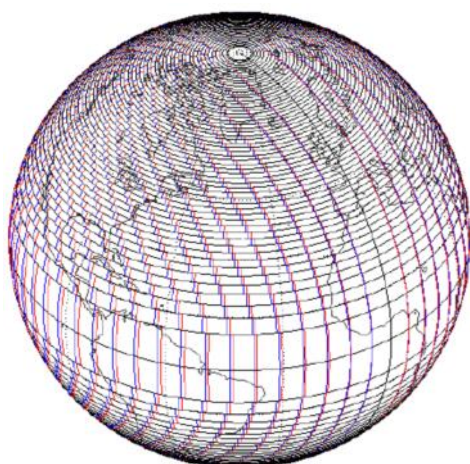


图4-2　经纬度区段划分

下面给出CPR编码的伪码描述。

```
Function [clat, clon] = cpr_encoding(lat, lon):
if even encoding
   i=0
elseif odd encoding
   i=1
NZ = 15;
Nb = 17; % only for airborne position
scalar = 2^Nb;
```

```
    dlat(i) = 360/(4*NZ-i);
    Yz(i) = round(scalar*mod(lat,dlat(i))/dlat(i));
    Rlat(i) = dlat(i)*(Yz(i)/scalar+ floor(lat/dlat(i)));
    If (NL(Rlat(i))-i)>0
        dlon(i) = 360/(NL(Rlat(i))-i);
    elseif (NL(Rlat(i))-i)=0
        dlon(i) = 360;
    Xz(i) = round(scalar*mod(lon,dlon(i))/dlon(i));
    clat = Yz1;
clon = Xz1;
```

4.2.5 CRC校验

CRC校验的基本原理请参考维基百科[8] 。

在ADS-B中使用的CRC校验的多项式为0Xfff409，二进制展开为POLY＝1111111111111010000001001，其生成多项式为：

$$G(x) = x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^{3} + 1$$

在发射端添加CRC校验比特的方法为：对前面的88bit除以多项式POLY，余数24bit就是CRC校验比特。

在接收端，对112bit除以多项式POLY，若余数为0（即错误图样为0），则可以判定解码得到的数据信息是正确的；否则，说明这112bit数据在传输过程中发生了错误。

4.3　ADS-B信号欺骗攻击

互联网上公开源码的ADS-B代码有很多，我们利用的是Nick Foster为GNU Radio写的Gr-air-modes[9]，这是一个ADS-B接收机程序。

这套代码使用的是GNU Radio的框架，可以支持UHD的USRP和osmocom的HackRF、bladeRF、RTLSDR等硬件。

ADS-B调制方式非常简单，因此在接收机算法方面没有太多可挖掘的地方，影响接收机性能最大的因素应该是射频硬件的指标和天线的指标。天线的位置也非常关键，要能够直接看到广阔的天空，遮挡越少越好。

为了产生虚假的ADS-B信号，使用Matlab程序产生了几个ADS-B报文，报文的内容是把4.2节中例子的经纬度替换为360公司大楼位置的经纬度，其余数据没有修改。Matlab程序处理了：

- CPR编码（具体参考4.2节中CPR编码的伪码描述）；
- CRC编码；
- 脉冲调制；
- 插入前导序列成帧。

将信号的采样率设为4MS/s。

使用两台计算机，其中计算机A把Matlab产生的数据文件用USRP发射出来，然后计算机B连接bladeRF，用modes_rx程序进行接收，如图4-3所示。

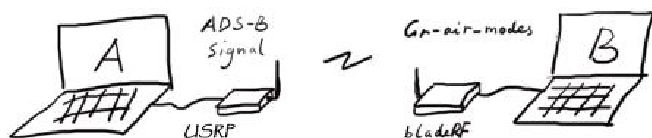


图4-3　ADS-B信号欺骗攻击实验

如图4-4所示计算机B接收到一个ADS-B信号，通过解析显示为一架飞机在360公司上空2175英尺高度盘旋。如图4-5所示，通过虚假的ADS-B信号，在Google Earth上显示出一架正在360公司上空盘旋的飞机。

```
alin@alin-desktop: ~/workspace/adsb
483337) at 2175ft
(-8 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.982315, 116.
484375) at 2175ft
(-9 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.983185, 116.
484192) at 2175ft
(-8 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.984009, 116.
482422) at 2175ft
^C(-9 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.983205, 11
6.482419) at 2175ft
(-9 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.982320, 116.
482482) at 2175ft
(-9 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.982315, 116.
483337) at 2175ft
(-8 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.982315, 116.
484375) at 2175ft
(-8 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.983185, 116.
484192) at 2175ft
(-8 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.983205, 116.
484167) at 2175ft
(-9 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.984009, 116.
482422) at 2175ft
(-7 0.00000000) Type 17 BDS0,5 (position report) from 75804b at (39.983205, 116.
482419) at 2175ft
alin@alin-desktop:~/workspace/adsb$
```

图4-4　接收到的ADS-B信号

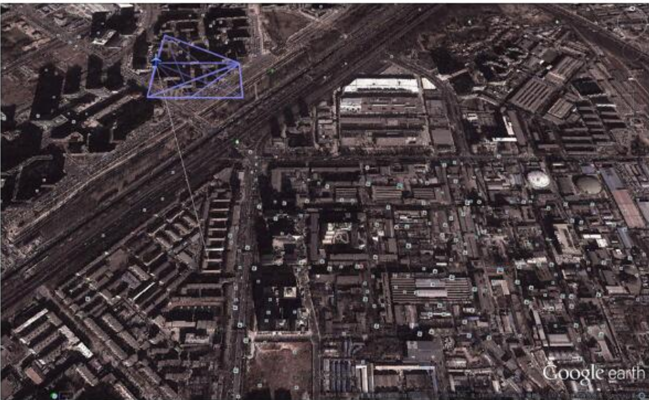


图4-5　在Google Earth上显示飞机的位置

4.4　攻防分析

构造虚假的ADS-B信号难度并不大，然而获取ADS-B信号的方式更加容易，利用廉价的无线电接收设备就可以接收飞机的ADS-B广播信息。接收S模式，与接收A/C模式的难易程度是一样的，只是S模式可以知道飞机的位置。

而要知道每一架民航客机的精确位置，更简单的方法是直接访问http://www.flightradar24.com等类似的网站，如图4-6所示。



图4-6　民航客机位置

既然有很多种简单的方法可以知道飞机的精确位置，那么为什么飞机还要大声地喊“我在这里”呢？因为这对于飞行安全来说非常重要，它意味着告诉地面“我在这里，我还活着”。因此ADS-B自动应答机是不允许关闭的，也是难以关闭的。

可是，在MH370事件中，ADS-B居然被关闭了，只剩下与海事卫星通信的ACARS系统留下的几个ping码信号。MH370事件给人们的教训是，需要更有力地保障监视飞机位置的能力。

所以，从这个角度来说，暴露飞机位置并不是一种风险，而是一种安全保障。

很显然，利用廉价的无线电设备广播ADS-B信号，可以恶意产生虚假的ADS-B目标。例如可以发射一个信号，表示有一架飞机正在越来越逼近另一架飞机，有撞机的可能，而实际上这是一架虚假的飞机。

但是，要让飞机或者地面雷达接收到这个虚假信号，并不是一件容易的事情。

若从地面发射虚假信号，发射功率要达到20W以上（未经仔细论证，仅做了简单网页搜索）。在关键的航空频段，这么大的发射功率是很容易被地面无线电监测设备发现的。若要在空中发射虚假信号，一种方法是携带一个小型无线电设备登机。要通过重重安检，并且在飞机上不被空乘人员发现，也不是一件容易的事情。

从发射信号的角度而言，不仅仅是ADS-B系统，所有的航空频段都面临着虚假信号的威胁，因此航空频段的无线电监测始终都是无线电监测部门的重点工作。例如机场和塔台之间的语音通话，频率就在118~135.975MHz范围内，使用的是最简单的AM调制。无线电爱好者可以使用这个波段的电台收听到机场通告，以及塔台与飞机之间的通话。同样，黑客也可以在这个频段发出干扰信号，当然也很容易被监测到。

而航空通信使用如此原始、简单的AM调制，同样是为了保障飞行安全，在极低信噪比的情况下，人耳比机器更容易分辨出飞行员呼叫的声音。

综上所述，ADS-B系统的安全风险是明显存在的，与其他航空通信系统的安全缺陷一样。但是，安全缺陷的存在是有其合理性的。目前来看，要保障航空通信的安全，只能从监测非法无线电信号和严格执行登机安检这两个方面来执行。

参考文献

[1]　李荣，译.1090MHz扩展断续振荡ADS-B最低工作性能标准（DO-260A）

[2]　http://adsb.tc.faa.gov/

[3]　RTCA Special Committee186，Working Group3ADS-B1090MOPS，Revision BMeeting#29，An Expanded Description of the CPR Algorithm，http://adsb.tc.faa.gov/WG3_Meetings/Meeting29/1090-WP29-07-Draft_CPR101_Appendix.pdf

[4]　RTCA Special Committee186，Working Group3ADS-B1090MOPS，Revision AMeeting#14，Proposed Revisions to Appendix Afor1090TIS-B and ADS-B MASPS Changes，http://www.anteni.net/adsb/Doc/1090-WP-14-09R1.pdf

[5]　http://www.lll.lu/~edward/edward/adsb/VerySimpleADSBreceiver.html

[6]　http://www.lll.lu/~edward/edward/adsb/DecodingADSBposition.html

[7]　Appendix AExtended Squitter and TIS-B Formats and Coding Definitions，http://adsb.tc.faa.gov/WG3_Meetings/Meeting30/1090-WP30-21-Appendix_A%20Mods.pdf

[8]　http://en.wikipedia.org/wiki/Cyclic_redundancy_check

[9]　https：//github.com/bistromath/gr-air-modes

[10]　RTCADO-181D，http://adsb.tc.faa.gov/SC209_Meetings/WG1_Mtg06-2008-0414/ModeS-WP06-08-DO-181D-v1-6.pdf

第5章　蓝牙安全

蓝牙是一种无线通信标准，工作在2.4～2.485GHz的ISM频段，用于短距离的数据连接，可以建立所谓的PAN（Personal Area Network）网络。蓝牙设备最常见的应用是各种以手机和电脑为中心的外围设备，例如与手机配合使用的蓝牙耳机、手机与汽车音响的互联、蓝牙键盘和鼠标等。本章将介绍与蓝牙系统相关的安全问题。

5.1 蓝牙技术简介

蓝牙技术最早是在1994年由爱立信发明的，最初的目的是用无线连接来代替RS-232线缆连接。

目前蓝牙标准由Bluetooth Special Interest Group（SIG）组织管理。这一组织有超过25000个成员公司，分布在电信、计算机、网络和消费电子领域。

“蓝牙”，即Bluetooth，是斯堪的纳维亚语言词汇Blatand/Blatann的英语化。这个词是10世纪时一位国王Harald Bluetooth的绰号。相传，他统一了若干丹麦部落成为一个王国，并引入了基督教。蓝牙技术的开发者Jim Kardach在1997年提出用Bluetooth这个名称，据说他当时正在读一本名为The Long Ships的小说，是讲述维京人和Harald Bluetooth国王的故事。他觉得蓝牙可以使各种手机与电脑相互通信，把各种不同的通信协议统一在一起，就像这位国王做的事情一样。

蓝牙的logo是符号文字ᚼ（Hagall）和ᛒ（Bjarkan）的组合，也就是国王名字Hagall Bjarkan的首字母。

蓝牙工作在2.4～2.485GHz频段，有79个频道，每个频道占据1MHz带宽。因为2.4GHz的ISM频段是非常繁忙的，为了对抗其他系统的干扰，蓝牙采用跳频式的扩频技术，通常1秒钟跳跃1600次。

BLE（Bluetooth Low Energy）只有40个频道，每个频道占据2MHz带宽。

蓝牙是一种基于数据包通信的协议，使用主-从式的网络结构。在一个piconet中，1个主节点可以与多达7个从节点通信。所有的设备都共享主节点的时钟。数据包以312.5μs的间隔互相交换，这个时间称为clock tick。两个clock tick构成一个625μs的时隙，两个时隙构成一个1250μs的时隙对。一种比较简单的时隙配置是，主节点在偶数时隙发送数据包，在奇数时隙接收数据包；从节点相反，在偶数时隙接收数据包，在奇数时隙发送数据包。这里讨论的是经典的蓝牙协议，BLE的空中接口协议与经典的蓝牙协议有很大的不同。

蓝牙协议还支持多个piconet连接在一起，组成scatternet。在这种情形下，某些设备可以在一个piconet中担任主节点，在另一个piconet中只是从节点。

蓝牙技术已经发展了很多年，蓝牙规范有很多个版本，从最早的蓝牙1.0到2014年发布的蓝牙4.2。早期的蓝牙标准只能支持几百Kbps的传输速率，而到了蓝牙3.0版本，已经可以支持最高24Mbps的传输速度。蓝牙4.0版本引入了BLE协议，即低功耗蓝牙，其功耗非常低，可以使设备的电池维持很长的时间，因此在很多可穿戴设备中得到了应用。

蓝牙与WiFi，有很多相似之处。WiFi主要用于更高速的无线连接，传输距离更远，称为WLAN（Wireless Local Area Network）；而蓝牙是WPAN（Wireless Personal Area Network）的概念。两者在很多应用场合有相互补充的作用。

WiFi以接入点为中心，是一种“客户端-服务器”结构，所有流量都通过接入点转发；而蓝牙的应用场景往往是两个设备之间点到点的连接，是对称的结构。蓝牙适合用在一些简单的应用场景中，两个设备之间的连接只需要非常简单的配置，例如按一下按键；而WiFi适合用在一些需要更复杂的连接配置的场景中。不过，实际上蓝牙协议本身也有接入点这样的中心式结构，而WiFi也有“WiFi Direct”这种直连方式，因此两者既互相补充，又互相模仿。

5.2 蓝牙安全概述

本节的介绍主要来源于维基百科的资料[注]。

蓝牙协议有加密和鉴权功能。蓝牙的密钥产生是基于蓝牙的PIN码的。PIN码需要输入到蓝牙设备中，或者出厂时就已经写入设备里。

在美国国家标准技术研究所官网上有一份蓝牙安全指南（Guide to Bluetooth Security[注]），介绍了如何安全地使用蓝牙技术。使用蓝牙技术非常的简单、易用，但也有明显的弱点，容易被拒绝服务攻击、窃听、中间人攻击、数据篡改、挪用资源。因此在选用蓝牙技术时，必须充分评估是否能够接受这样的风险。这份指南同时也给出了一些指导，在创建和维护一个蓝牙系统时，如何去降低它的安全风险。

2001年，贝尔实验室的Jakobsson和Wetzel发现了蓝牙配对协议中的缺陷，以及加密方法中的漏洞。2003年，A.L.Digital公司的两位研究人员Ben和Adam Laurie发现，在一些蓝牙产品中有严重的漏洞，能够导致个人隐私数据的泄露。于是，接下来就出现了一种利用这一漏洞的BlueBug攻击手法，开始展现出这一漏洞的危害性。

2004年，第一个所谓的蓝牙病毒出现，它可以在Symbian操作系统的手机中传播。这个“病毒”是一个名叫“29A”的团队编写的PoC程序，写出来之后就交给了反病毒机构，因此它具有威胁性，但并没有真正地传播过。2004年8月，一项实验创纪录地把蓝牙连接的距离增加到了1.78公里，在实验中使用了方向性天线和功率放大器。这就增加了蓝牙病毒的危险性，使它能够威胁到距离非常远的蓝牙设备。

2005年1月，出现了一种名为“Lasco.A”的手机蠕虫病毒，也是针对Symbian手机的。用户一旦通过蓝牙接收这个病毒文件，病毒就会自动地安装在系统中，然后开始寻找周围其他的蓝牙设备去感染。另外，这种病毒还能够感染系统中其他的SIS文件。最终的后果是导致手机系统不稳定。

2005年4月，剑桥大学的安全研究人员公布了他们发现的一种被动式攻击方式，能够攻击基于PIN码的配对协议。这种攻击方式操作起来非常快，证实了蓝牙的对称密钥的建立方法是有缺陷的。为了抵

御这种攻击，研究人员提出了一种更强的非对称密钥加密方法[注]。

2005年6月，Yaniv Shaked和Avishai Wool发表了一篇论文，提出了被动式和主动式两种攻击方式，可以获得PIN码。在被动式攻击中，攻击者在蓝牙设备初始配对的时刻就能窃听到通信的内容；而主动式攻击则需要在某个特定的时刻发出一种特殊构造的数据包，使主节点和从节点重复配对过程，之后就可以使用被动式攻击在配对过程中侦听密钥了。这种攻击的主要弱点在于，它需要受到攻击的用户在攻击期间重新输入PIN码，此时设备会有提示。另外，这种攻击还需要一些特别的设备在恰当的时刻发出一种特殊构造的数据包，现成的蓝牙芯片是不能做到这一点的。

蓝牙地址还可以用于追踪设备的位置。2005年8月，英格兰剑桥郡的警方发出警示说，有些小偷利用蓝牙来探查车内是否有被遗忘的手机或电脑等设备，提示人们在不使用蓝牙功能时尽量关闭它。

2006年4月，来自Secure Network和F-Secure的研究人员发布了一份报告，警告有大量留在“可见”状态的设备，还报告了各种蓝牙服务的普及程度，以及与蓝牙蠕虫传播的危险性有关的统计数据。

2007年10月，在卢森堡的Hack.lu Security会议上，Kevin Finistere和Thierry Zoller演示了通过蓝牙远程root Mac OS Xv10.3.9和v10.4系统，还演示了一个蓝牙PIN码和Linkeys破解器。

维基百科上介绍的以上这些蓝牙安全的案例都至少是9年以前的事情了。那么近些年来，在蓝牙安全领域有哪些新的东西出现呢？这些新内容将在接下来的章节中展开介绍。

1. https：//en.wikipedia.org/wiki/Bluetooth返回

2. http://csrc.nist.gov/publications/drafts/800-121r1/Draft-SP800-121_Rev1.pdf返回

3. https：//www.cl.cam.ac.uk/~fms27/papers/2005-WongStaClu-bluetooth.pdf返回

5.3　蓝牙嗅探工具Ubertooth

如图5-1所示，Ubertooth是用于蓝牙嗅探的开源硬件，是无线硬件黑客Michael Ossmann带领的团队做的。2010年10月，Michael Ossmann在ToorCon12会议上展示了他们开发出的第一代Ubertooth Zero。新一代的Ubertooth One是在2011年完成的。因为Ubertooth是开源硬件，所以有很多售卖的地方，价格大约为120美元。

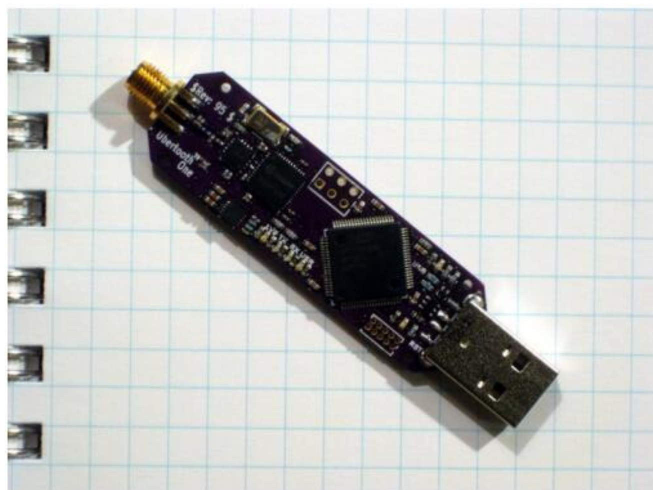


图5-1　Ubertooth

5.3.1 Ubertooth软件安装

Ubertooth的源码放在GitHub上，地址是https：//github.com/greatscottgadgets/ubertooth。

在安装Ubertooth软件之前，首先要安装一些依赖库。以Ubuntu系统为例，安装以下依赖库：

```
sudo apt-get install cmake libusb-1.0-0-dev make gcc g++ libbluetooth-dev \
pkg-config libpcap-dev python-numpy python-pyside python-qt4
```

接下来是编译安装libbtbb，这是一个蓝牙基带协议的程序。源码也是从GitHub上下载的。

```
wget https://github.com/greatscottgadgets/libbtbb/archive/2015-10-R1.tar.gz -O libbtbb
tar xf libbtbb-2015-10-R1.tar.gz
cd libbtbb-2015-10-R1
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

安装用于蓝牙嗅探的工具软件：

```
wget https://github.com/greatscottgadgets/ubertooth/releases/download/2015-10-R1/
ubertooth-2015-10-R1.tar.xz -O ubertooth-2015-10-R1.tar.xz
tar xf ubertooth-2015-10-R1.tar.xz
cd ubertooth-2015-10-R1/host
mkdir build
cd build
cmake ..
make
sudo make install
```

插上硬件，升级固件。进入ubertooth-one-firmware-bin目录：

```
$ ubertooth-dfu -d bluetooth_rxtx.dfu -r
Checking firmware signature
```

```
No DFU devices found - attempting to find Ubertooth devices

1) Found 'Ubertooth One' with address 0x1d50 0x6002

Select a device to flash (default:1, exit:0):
```

输入回车之后，Ubertooth会自动进入DFU模式，然后开始写入固件。写完之后，只需要拔插一下Ubertooth，就会回到正常操作模式。

5.3.2 使用Ubertooth

Ubertooth主要是一个开发平台，可以编写合适的代码使用这个硬件来满足自己的需求，这样才能最大地发挥它的作用。如果你刚刚拿到Ubertooth，则可以先使用这个硬件中一些现成的小工具来做一些简单的实验。

1.频谱分析

最简单的一个实验就是用Ubertooth观察频谱。把Ubertooth插在电脑上，接上一个天线，这时你会看到RST和1V8这两个灯亮起来。RST灯表示LPC175x微控制器已经在运行，1V8灯表示CC2400芯片已经上电。如果使用的是Linux系统，USB灯也会亮起来。

进入host/python/specan_ui目录，运行“ubertooth-specan-ui”，你就可以看到2.4GHz频段的信号频谱，如图5-2所示。

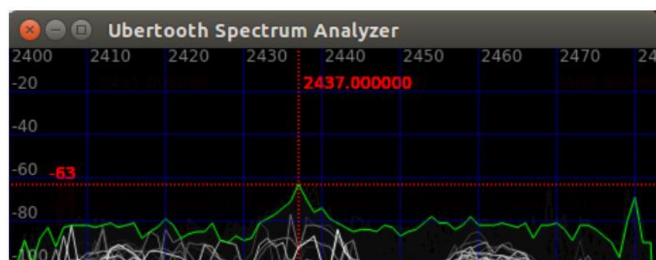


图5-2　使用Ubertooth观察频谱

此时RX灯会亮起来，表示Ubertooth正处于接收模式。

2.侦听LAP地址

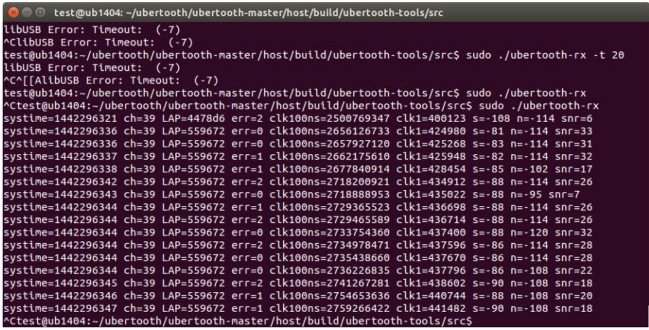
蓝牙数据包的开头部分是LAP（Lower Address Part）地址，LAP地址是蓝牙设备地址（BD_ADDR）的低位部分。BD_ADDR是48比特的地址，LAP是BD_ADDR的低位部分的24比特。LAP地址在每一个数

据包中都会被发送，因此最重要的一种被动侦听方式就是侦听LAP地址，通过LAP地址就可以知道附近有哪些蓝牙设备。

在安装Ubertooth软件的过程中，已经把一些可执行程序安装到了系统目录下，因此可以直接运行：

```
sudo ubertooth-rx
```

可以看到抓到了一些数据包，侦听到的LAP地址主要是559672，如图5-3所示。



```
test@ub1404: ~/ubertooth/ubertooth-master/host/build/ubertooth-tools/src
libUSB Error: Timeout:  (-7)
^ClibUSB Error: Timeout:  (-7)
test@ub1404:~/ubertooth/ubertooth-master/host/build/ubertooth-tools/src$ sudo ./ubertooth-rx -t 20
libUSB Error: Timeout:  (-7)
^C^[[AlibUSB Error: Timeout:  (-7)
test@ub1404:~/ubertooth/ubertooth-master/host/build/ubertooth-tools/src$ sudo ./ubertooth-rx
^Ctest@ub1404:~/ubertooth/ubertooth-master/host/build/ubertooth-tools/src$ sudo ./ubertooth-rx
systime=1442296321 ch=39 LAP=4478d6 err=2 clk100ns=2500769347 clk1=400123 s=-108 n=-114 snr=6
systime=1442296336 ch=39 LAP=559672 err=0 clk100ns=2656126733 clk1=424980 s=-81 n=-114 snr=33
systime=1442296336 ch=39 LAP=559672 err=0 clk100ns=2657927120 clk1=425268 s=-83 n=-114 snr=31
systime=1442296337 ch=39 LAP=559672 err=1 clk100ns=2662175610 clk1=425948 s=-82 n=-114 snr=32
systime=1442296338 ch=39 LAP=559672 err=1 clk100ns=2677840914 clk1=428454 s=-85 n=-102 snr=17
systime=1442296342 ch=39 LAP=559672 err=2 clk100ns=2718200921 clk1=434912 s=-88 n=-114 snr=26
systime=1442296343 ch=39 LAP=559672 err=0 clk100ns=2718888953 clk1=435022 s=-88 n=-95 snr=7
systime=1442296344 ch=39 LAP=559672 err=1 clk100ns=2729365523 clk1=436698 s=-88 n=-114 snr=26
systime=1442296344 ch=39 LAP=559672 err=2 clk100ns=2729465589 clk1=436714 s=-88 n=-114 snr=26
systime=1442296344 ch=39 LAP=559672 err=0 clk100ns=2733754360 clk1=437400 s=-88 n=-120 snr=32
systime=1442296344 ch=39 LAP=559672 err=2 clk100ns=2734978471 clk1=437596 s=-86 n=-114 snr=28
systime=1442296344 ch=39 LAP=559672 err=2 clk100ns=2735438660 clk1=437670 s=-86 n=-114 snr=28
systime=1442296344 ch=39 LAP=559672 err=0 clk100ns=2736226835 clk1=437796 s=-86 n=-108 snr=22
systime=1442296345 ch=39 LAP=559672 err=2 clk100ns=2741267281 clk1=438602 s=-90 n=-108 snr=18
systime=1442296346 ch=39 LAP=559672 err=1 clk100ns=2754653636 clk1=440744 s=-88 n=-108 snr=20
systime=1442296347 ch=39 LAP=559672 err=1 clk100ns=2759266422 clk1=441482 s=-90 n=-108 snr=18
^Ctest@ub1404:~/ubertooth/ubertooth-master/host/build/ubertooth-tools/src$
```

图5-3　使用Ubertooth侦听LAP地址

3.Kismet配合Ubertooth使用

更高级一些的蓝牙侦听，可以通过Kismet实现。Kismet是侦听802.11系统的一个非常著名的软件，Ubertooth为Kismet写了一个插件，于是就可以用Kismet来分析蓝牙数据包了，如图5-4所示。

Kismet Sort View Windows

BD_ADDR	Count
00:00:00:EE:1C:14	10
00:00:BE:FA:6A:F6	11
00:00:00:8D:B4:0F	12
00:00:00:C5:20:EC	12
00:00:00:76:9D:95	13
00:00:00:FE:4D:E3	13
00:00:00:86:FA:DF	14
00:00:44:83:40:D3	15
00:00:95:CB:98:B4	15
00:00:00:81:97:20	15
00:00:58:90:D8:02	15
00:00:00:08:57:52	16
00:00:D8:C9:DE:97	17

图5-4　使用Kismet＋Ubertooth侦听蓝牙数据包

在host/kismet/plugin-ubertooth/README中，可以看到Kismet插件的使用指导。

Kismet不仅标记出LAP地址，还标记出更高的8比特地址（UAP），这是通过分析多个数据包解码得到的。Kismet的另一个好处是，它可以把解码后的数据包保存为pcapbtbb文件，于是我们就可以用Wireshark来读取了，便于分析。

一些商用的WiFi监控产品往往也有蓝牙监控功能，就是通过类似于Kismet＋Ubertooth这样的方式来实现的。

5.4 低功耗蓝牙

BLE（Bluetooth Low Energy，低功耗蓝牙）是蓝牙4.0版本中的一部分。现在BLE技术在各种可穿戴设备中得到大量的应用，比如各种手环、小挂件，还有iBeacon类的定位标记，都在使用BLE芯片。因此在智能硬件破解中，常常可以看到关于BLE破解的例子。

BLE与传统的蓝牙在协议上有很大的不同。Ubertooth也支持BLE的侦听，但是更常用的侦听工具是BLE Sniffer，它是用CC2540USB Dongle来做的。

5.4.1　TI BLE Sniffer

BLE Sniffer软件可以在TI的官网上找到，地址是http://processors.wiki.ti.com/index.php/BLE_sniffer_guide。

CC2540USB Dongle的样子如图5-5所示。

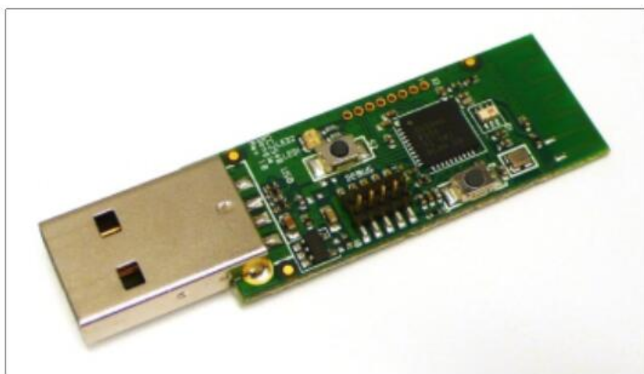


图5-5　CC2540 USB Dongle

下载软件并且安装好之后（Windows环境），插上Dongle，打开软件，如图5-6所示。

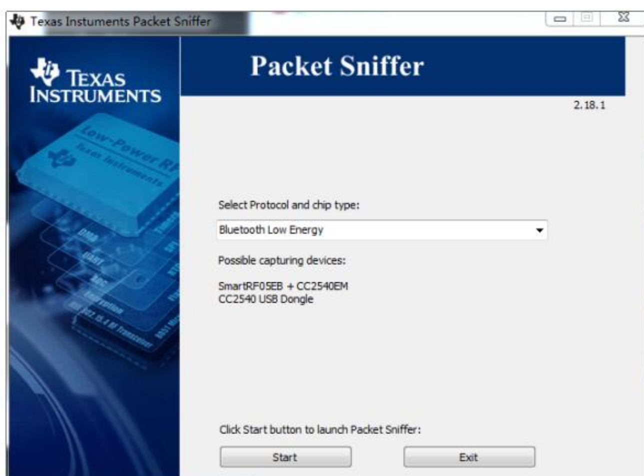


图5-6　Packet Sniffer启动界面

单击“Start”按钮，就可以进入到Sniffer界面，如图5-7所示。

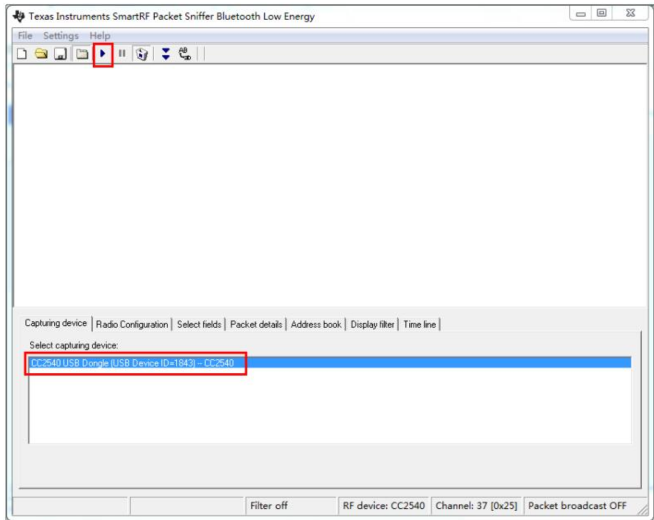


图5-7　Packet Sniffer主界面

选择这个Dongle设备，单击“Play”按钮，就可以侦听到BLE的数据包。例如，在图5-8中，可以看到很多广播数据包，它们都是在37号信道上侦听到的数据包。



图5-8　Packet Sniffer侦听到的广播数据包

有很多BLE设备都是这样不停地广播自己的，iBeacon就是一个典型的例子。商家可以在自己的店铺里放置iBeacon设备，顾客的手机

侦听到这个广播之后，就可以连接上来获得一些推送的广告信息，如图5-9所示。

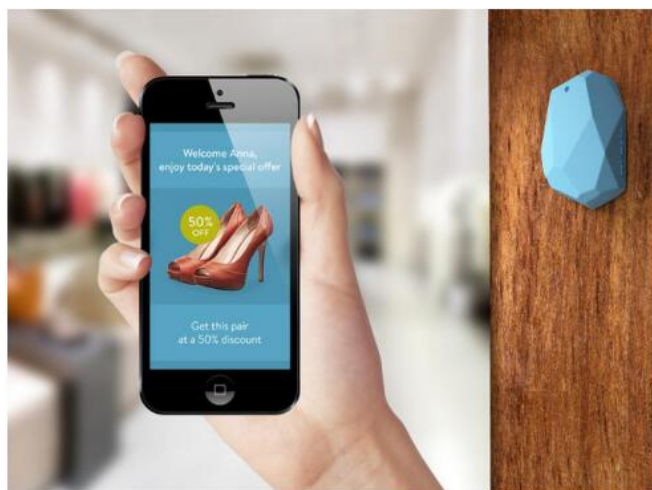


图5-9　Estimote公司的iBeacon设备用在商店里做广告推送

除了广播包之外，Packet Sniffer还可以侦听到其他的数据包。但是因为USB Dongle只有一个射频模块，在任意时刻只能在一个信道上侦听，而蓝牙通信又是跳频的，所以它不一定能跟踪到这个数据包。

如果在侦听时恰好遇到两个蓝牙设备正在建立连接，那么Packet Sniffer会自动跟踪这个连接的跳频节奏，跟着这个连接的频率跳，于是可以侦听到这个连接所有的数据包。看下面这个例子，如图5-10所示。

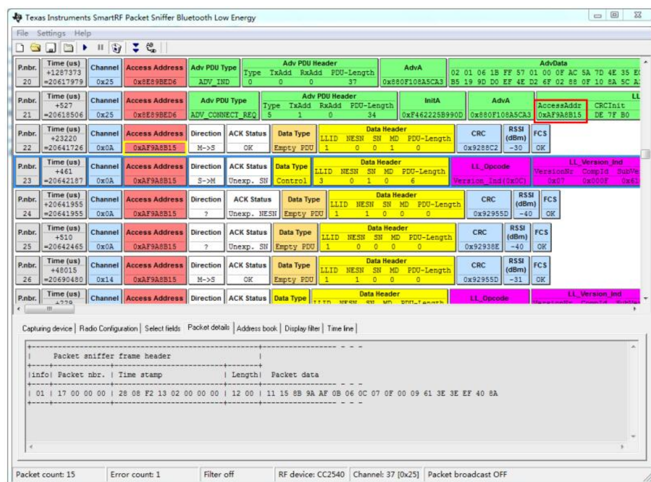


图5-10　使用Packet Sniffer观察小米手环与手机的连接建立过程

可以看到Packet Sniffer侦听到了一个ADV_CONNECT_REQ类型的包，其中给出了AccessAddr＝0xAF9A8B15，接下来两个BLE设备就可以使用这个接入地址相互通信了。

连接建立的过程是很关键的，一些密钥会在这个过程中传递，因此很多破解智能硬件的案例，都使用Packet Sniffer来分析连接建立的流程。

5.4.2　使用手机应用读写BLE设备的属性

手机本身有BLE芯片，因此有一些手机App能够连接周围的BLE设备，还能够对某些属性进行读写操作。例如苹果手机有一个应用叫LightBlue，如图5-11所示。



图5-11　LightBlue App

从App的简介可以看出，它可以用来“发现”周围的BLE设备，还可以建立连接。乌云有一篇文章《物联网安全拔“牙”实战—低功耗蓝牙（BLE）初探》注，介绍了如何用LightBlue使小米手环、跳蛋等智能硬件振动。

在这里拿小米手环做例子，当LightBlue连接上小米手环之后，可以看到一个名为FEE7的UUID，如图5-12所示。

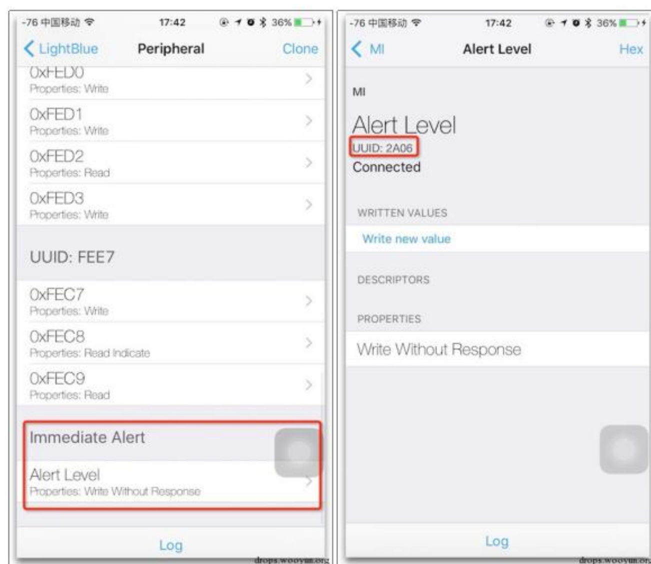


图5-12　使用LightBlue使小米手环振动

其中的0xFE**就是私有Characteristic的UUID，下面的“Immediate Alert”显示出了名称，表示其不是小米私有的Service，而是官方公开定义的Service。点击进入Characteristic，看到这个Characteristic的UUID为2A06。在BLE的规范中，规定了Characteristic的几个值的含义：0，无警告；1，温和的警告；2，强烈的警告；3~255，预留。点击“Write new value”，可以写入新的值，如果写入1或2，就可以引起手环的振动。

除了“Immediate Alert”以外，还有很多其他的属性，在遇到私有Service或Characteristic时，就要通过App逆向或嗅探蓝牙通信来分析了。

但是通过LightBlue修改BLE的属性的前提条件是，LightBlue要能够连接上攻击目标。一般来说，如果攻击目标没有初始化，或者正处于连接状态，那么攻击者要想把连接抢过来是比较困难的。

1. http://drops.wooyun.org/tips/10109返回

5.4.3　模拟BLE设备发射数据包

本节首先介绍焦现军博士写的一个BLE发包工具，代码在GitHub上，地址为https：//github.com/JiaoXianjun/BTLE。

这个项目中不仅有发包程序，也有收包程序，收包程序可以实现类似于TI的Packet Sniffer的功能。我们主要来看一下发包程序。

下载源代码之后，编译程序。前提是你已经安装好了HackRF或者bladeRF的驱动，这个程序目前只支持这两款硬件。

编译很简单，如下所示：

```
cd host
mkdir build
cd build
cmake ../      （不使用参数-DUSE_BLADERF＝1的话，则表示默认使用HACKRF）
make
sudo make install  （也可以不安装，直接使用btle-tools/src目录下的btle_tx）
```

根据网页上readme的介绍，发送一个iBeacon信号：

```
btle_tx    37-iBeacon-AdvA-010203040506-UUID-B9407F30F5F8466EAFF925556B57FE6
```

其中的参数含义如下：

- 37，信道37，在37号广播信道发射。
- iBeacon，数据包格式是iBeacon格式。
- AdvA，广播地址，设为010203040506。
- UUID，设为Estimote产品固定的UUID：B9407F30F5F8466EAFF925556B57FE6D。
- Major，iBeacon格式中的major参数，设为0008。
- Minor，iBeacon格式中的minor参数，设为0009。

- TxPower，发送功率设为C5。
- Space，发送数据包的时间间隔，设为100ms。
- r100，重复发送100次。

程序运行起来是这个样子，如图5-13所示。

此时，可以用一个带有BLE功能的手机进行查看。苹果手机可以使用Locate Beacon App，如图5-14所示。

```
test@ub1404: ~/workspace/BTLE-master/host/build/btle-tools/src

test@ub1404:~/workspace/BTLE-master/host/build/btle-tools/src$ ./btle_tx     37-
iBeacon-AdvA-010203040506-UUID-B9407F30F5F8466EAFF925556B57FE6D-Major-0008-Minor
-0009-TxPower-C5-Space-100     r100
num_repeat 100
num_packet 1

packet 0
channel_number 37
pkt_type IBEACON
payload_len 36
num_info_bit 344
after crc24
aad6be898e4024060504030201020 11a1aff4c000215b9407f30f5f8466eaff925556b57fe6d0008
0009c54e7bb0
after scramble 368 0
aad6be898ecdf651a439a464b177300b52693bf8e15350ebafaea6cb9ed437f1019e50ab8fcef45d
68c66c5717ed
num_phy_bit 368
num_phy_sample 1488
space 100
INFO bit:aad6be898e402406050403020102011a1aff4c000215b9407f30f5f8466eaff925556b5
7fe6d00080009c5
 PHY bit:aad6be898ecdf651a439a464b177300b52693bf8e15350ebafaea6cb9ed437f1019e50a
b8fcef45d68c66c5717ed
PHY SMPL: PHY_bit_for_matlab.txt IQ_sample_for_matlab.txt IQ_sample.txt IQ_sampl
e_byte.txt

r0 p0 at 0us
r1 p0 at 319276us
r2 p0 at 317045us
r3 p0 at 319934us
r4 p0 at 318497us
r5 p0 at 319128us
r6 p0 at 317706us
```

图5-13　使用btle_tx程序通过HackRF发送iBeacon信号

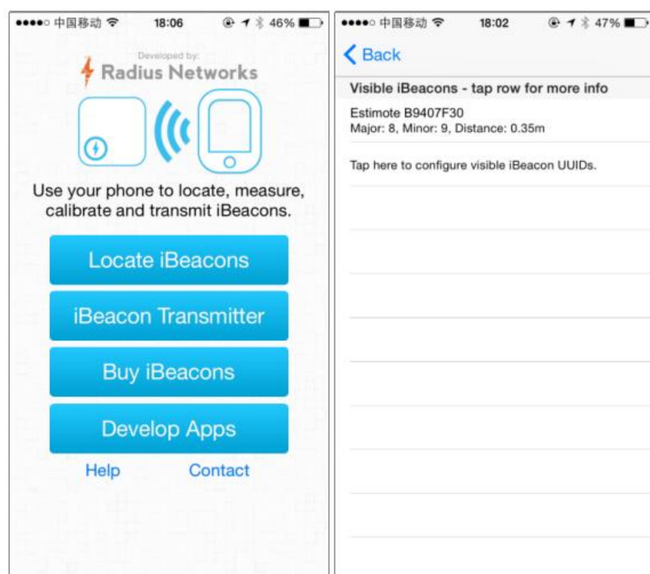
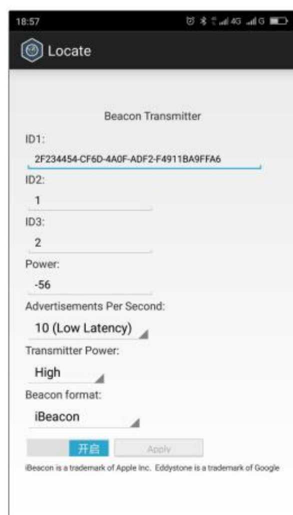


图5-14　使用手机观察通过HackRF发送的iBeacon信号

可以在手机上看到附近有一个名为“Estimote”、ID是B9407F30的iBeacon，跟我们发送的信号一致。

实际上，更简单的发送Beacon信号的方法就是直接用手机。例如上面的Locate Beacon App就有Beacon Transmitter的功能，我们用一台安卓手机做Beacon发射机，这台苹果手机做Beacon接收机，如图5-15所示。



手机A发送Beacon信号



手机B接收Beacon信号

图5-15　使用手机发送Beacon信号

使用这种方法就可以模拟一个Beacon信号，由此可见，模拟一个Beacon信号是很容易的，它只是一种广播信号，没有任何校验。

曾经发生过一件有意思的事情：2014CES大会设计了一个寻宝活动，让参观者在会场中的各个地方搜集iBeacon，最后可以获得奖品。然而，在会议开始之前，Make Magazine的一个团队就破解了游戏的安卓APK，找到了所有的Beacon Profile，于是无须进入会场，就可以模拟所有的Beacon，赢得游戏。

由此可见，Beacon这种广播应用只适合用在对安全性要求不高的场合。后来各个Beacon运营商/开发者使用了其他一些方法来增强Beacon的安全性，在此就不赘述了。从苹果的iBeacon来看，自2013年推出之后，到2014年就已经受到很多质疑，认为这种技术颇为“鸡肋”。今后是否渐渐地销声匿迹，只能待时间给出答案了。

第6章　ZigBee安全

本章的研究对象是ZigBee网络，这是一种主要使用在物联网中、输距离不太远的通信技术。

6.1 ZigBee简介

ZigBee是一个标准，它定义了一套低速率、低功耗的无线网络协议。基于ZigBee的设备工作于868Mhz、915Mhz以及2.4Ghz频段，最大传输速率为250Kb/s。ZigBee主要针对那些由电池供电的低功耗、低速率、低成本的应用。在典型的应用如智能家居、传感器网络等中，ZigBee设备大部分时间（99%以上）都工作在低功耗也叫休眠状态，所以ZigBee设备可以连续工作几年而不用换电池。

ZigBee标准是由ZigBee联盟定义的，ZigBee联盟由许多来自半导体、软件开发及设备制造等行业的几百家公司组成。ZigBee联盟主席Bob Heile通过如下故事解释了ZigBee的来源：

在挪威传说中有一个名叫ZigBee的妖怪，住在松恩峡湾一个岛上的村子里。与其他挪威传说中那些巨大可恶的妖怪不同，ZigBee是一个友好的小妖怪，它很少说话，但是它说的话都非常可靠，人们可以信任它。

一次，ZigBee感应到堆在粮仓旁边的一堆稻草由于腐烂发热开始自燃，它立即向村子里的每一户发起警报，村民们得以及时地将火扑灭保住了粮仓。

还有一次，在一个寒冬过后的暖春，一个名叫Haarold Bluetooth的村民在村子附近的雪山上放羊，他想将羊群赶到他非常熟悉的小溪边去喝水，由于是暖春，快速融化的雪水已经将小溪变成了波涛汹涌的大河，意识到危险的Bluetooth想赶在洪水到达村庄之前通知村民，但是他离村庄太远，他的声音村民们根本就听不见。而此时的ZigBee也感到了危险，因为它也看到了洪水，它也和Bluetooth一样，无法一下子将声音传到村子里去，所以它立即开始往山下跳，就这样跳啊跳啊直到跳到村子里，它还自动打开了大坝的水闸泄掉了洪水，保住了村子。

村民们都很庆幸有ZigBee，因为不像Bluetooth，ZigBee能够一步步跳回村子。

以上故事引用自ZigBee wireless Networking，作者Drew Gislason。

短短一个故事把ZigBee的典型应用场景（传感器网络、自动控制及自然灾害预警等）、工作特征（大部分时间都工作在休眠状态，只有很少的时间用于通信，数据传输可靠）、协议特点（体积小，资源要求不高）以及与蓝牙的区别（多跳自组织）展现得淋漓尽致。

一个典型的ZigBee应用是病人监护，病人佩戴的血压计和心率计以固定周期采集血压和心率，然后通过ZigBee将数据传送到家里的PC进行初步分析，最后通过互联网将这些数据传送到病人的医生那里进行进一步分析。

6.1.1　ZigBee与IEEE802.15.4的关系

下面利用经典的OSI分层模型来介绍ZigBee与IEEE802.15.4的关系。

如图6-1所示，底部的媒介访问层和物理层由IEEE802.15.4标准定义，而IEEE802.15.4由IEEE802标准委员会定义。ZigBee标准只定义了网络层、应用层以及安全层，而采用了现成的IEEE802.15.4的MAC层及物理层作为ZigBee网络协议的一部分。

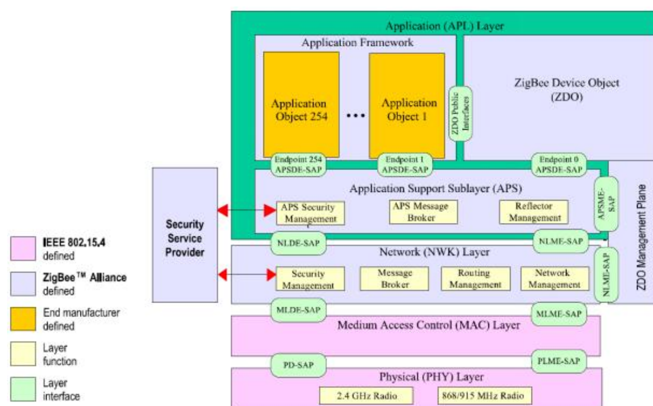


图6-1　ZigBee无线网络协议层次结构图[注]

1. 此图引用自ZigBee wireless networks and transceivers，作者Shahin Farahani。返回

6.1.2　802.15.4帧结构

ZigBee物理层由IEEE802.15.4标准定义，下面用ZigBee代替802.15.4来称呼它。

ZigBee物理层可以工作在868MHz、915MHz、2.4GHz三个频段，这些频段又被划分为27个信道（channel），如表6-1所示。

表6-1 ZigBee物理层工作频段划分信道

信道	信道宽度	频率范围	速率
0	600kHz	869~868.6MHz	100Kbps
1~10	2MHz	902~928MHz	250Kbps
11~26	5MHz	2.4~2.485MHz	250Kbps

与IEEE802.11类似，ZigBee使用直序扩频。与蓝牙不同，ZigBee通信如果不被重新配置将保持在同一个信道上而不会跳频，所以要对ZigBee网络的数据进行抓取就相对简单一些。

ZigBee数据和命令都是通过数据帧来传输的，应用层的数据被逐层打包，最后被物理层发送到目标节点。如图6-2所示为ZigBee数据帧结构图。

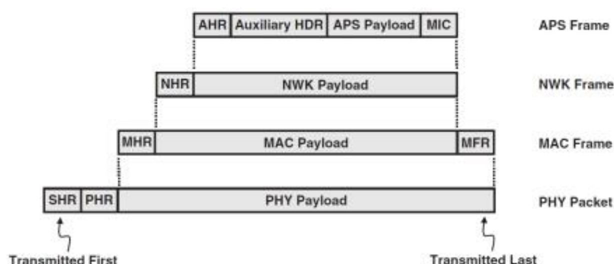


图6-2　ZigBee数据帧结构图注

可以看出，ZigBee数据在发送时从上到下层层打包，每层都加上相应的帧头及控制信息，而在接收时又从下到上逐层去掉头部及附加信息，最终得到应用数据。

- 物理层数据帧（PHY Packet），同步头（Synchronization Header，SHR）可以让数据接收方与发送方进行同步，从而锁定数据

流，物理层报头（Physical Layer Header，PHR）包含帧长度信息，物理层载荷（PHY Payload）数据是由上层提供的，包含了要传送的数据和命令。

- MAC层帧（MAC Frame），它是被作为物理层的载荷来传输的，其包含三个部分，MAC头（MHR）、MAC载荷（MAC Payload）和MAC尾（MFR），其中MHR包含了MAC层寻址及安全信息，MAC载荷包含了网络层数据帧，MFR包含了16位CRC帧校验。

- 网络层帧（NWK Frame），由头部（NHR）和载荷（NWK Payload）组成，其中头部包含了网络层寻址及安全信息，载荷包含了应用支持子层（Application Support Sublayer，APS）帧。在APS帧中，帧头AHR包含了应用层寻址及控制信息，辅助帧头（Auxiliary HDR）包含了密钥序号等安全相关信息。网络层和MAC层也有可选辅助帧头用于增强安全性。应用支持子层载荷（APS Payload）中包含了应用数据或命令。消息完整性验证码（Message Integrity Code，MIC）用于检查消息是否被非法篡改，因为在生成MIC时用到了密钥，所以如果没有密钥是无法对消息进行篡改而不被发现的。

1. 此图引用自ZigBee wireless networks and transceivers，作者Shahin Farahani。返回

6.1.3 ZigBee的MAC帧类型

与WiFi等其他无线协议不同，ZigBee只用了如下几种MAC层数据帧来传输数据。

- 信标帧（Beacon Frame）：用于网络扫描，当一个新节点想加入网络时，先发送信标请求（Beacon Request），收到信标请求的节点发送自己的信标，新节点收到信标后就可以知道这个网络的存在。

- 数据帧（Data Frame）：用于交换任意数据，最大长度为114字节。

- 响应帧（Acknowledge Frame）：数据发送方可以要求在接收方收到数据后发送一个响应帧，来通知自己对方已经收到数据。

- 命令帧（Command Frame）：类似于802.11标准中的网络管理帧（Management Frame），负责控制如关联、去关联、网络地址冲突以及缓存数据传送等。

6.1.4 ZigBee设备类型及网络拓扑

在IEEE802.15.4网络中按照设备能力分为两种设备：全功能设备（Full Function Device，FFD）和部分功能设备（Reduced Function Device，RFD），FFD可以完成IEEE802.15.4标准中规定的所有工作，而RFD则只能完成部分功能。例如，FFD可以和网络中的所有设备通信，而RFD则只能和FFD通信，RFD用在一些简单的应用中，如一个开关。

在IEEE802.15.4网络中按照设备所扮演的角色分为三种设备，即PAN（Personal Area Network）协调器（Coordinator）、协调器和普通设备。协调器是一个FFD，可以对网络中的信息进行中继，如果某协调器又是一个PAN的主控制器，则这个协调器称为PAN协调器，普通设备则是非协调器的其他设备。

在ZigBee的术语中，同样的角色有不同的名称，ZigBee的协调器就是IEEE802.15.4中的PAN协调器，ZigBee的路由器（Router）就是IEEE802.15.4中的协调器，ZigBee的终端节点（End Device）就是IEEE802.15.4中的普通设备，如图6-3所示。

- 协调器：全功能设备，负责控制整个网络和对消息进行中继，还负责对加入网络的节点进行鉴权，可以拒绝节点加入。

- 路由器：全功能设备，负责对数据包进行中继转发，可以与协调器和终端节点通信。

- 终端节点：部分功能设备，不能转发数据，不能和终端节点通信，只能和路由器及协调器通信。

ZigBee组网是由网络层来管理的，并且有三种网络拓扑，即星型（Star）、网状型（Mesh）和树型（Tree），如图6-4所示。

尽管每个ZigBee网络都需要一个协调器来管理整个网络，但是网络拓扑还决定是否需要路由器来对数据进行中继转发。

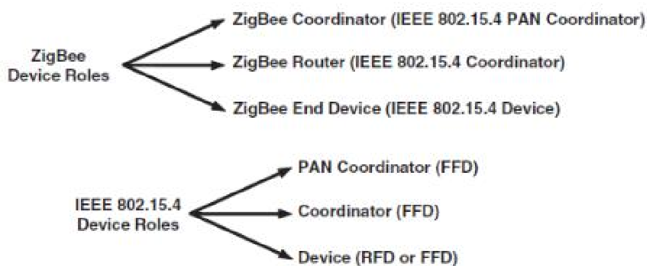


图6-3　ZigBee及IEEE802.15.4中的设备角色注

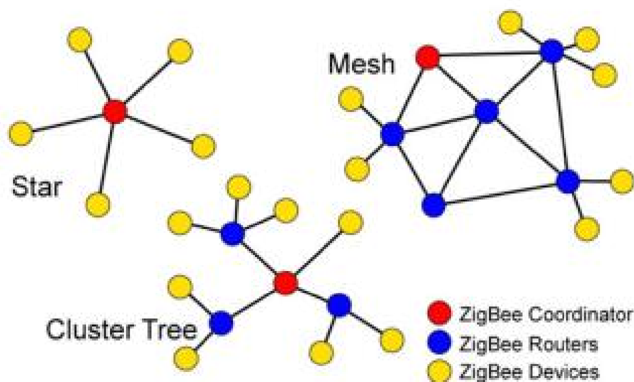


图6-4　ZigBee网络拓扑图注

1. 此图引用自ZigBee wireless networks and transceivers，作者Shahin Farahani。返回

2. 此图引用自http://www.efxkits.co.uk/role-of-2-4-ghz-wireless-communication-transceivers-in-communication/。返回

6.1.5 ZigBee组网过程

ZigBee组网由网络层（NWK Layer）负责，网络层还负责设备发现、网络地址分配、路由等。组网过程如下：

一个协调器（也是FFD）通过设备发现（利用前面提到的信标帧）来扫描本信道（预先配置好的）上工作的所有网络，然后随机选择一个与已有网络地址不同的网络地址，最后接收来自路由器节点及终端节点发起的入网请求，当一个节点加入网络时，协调器给它分配一个16位的网络地址。如图6-5所示为新节点入网过程的通信数据。

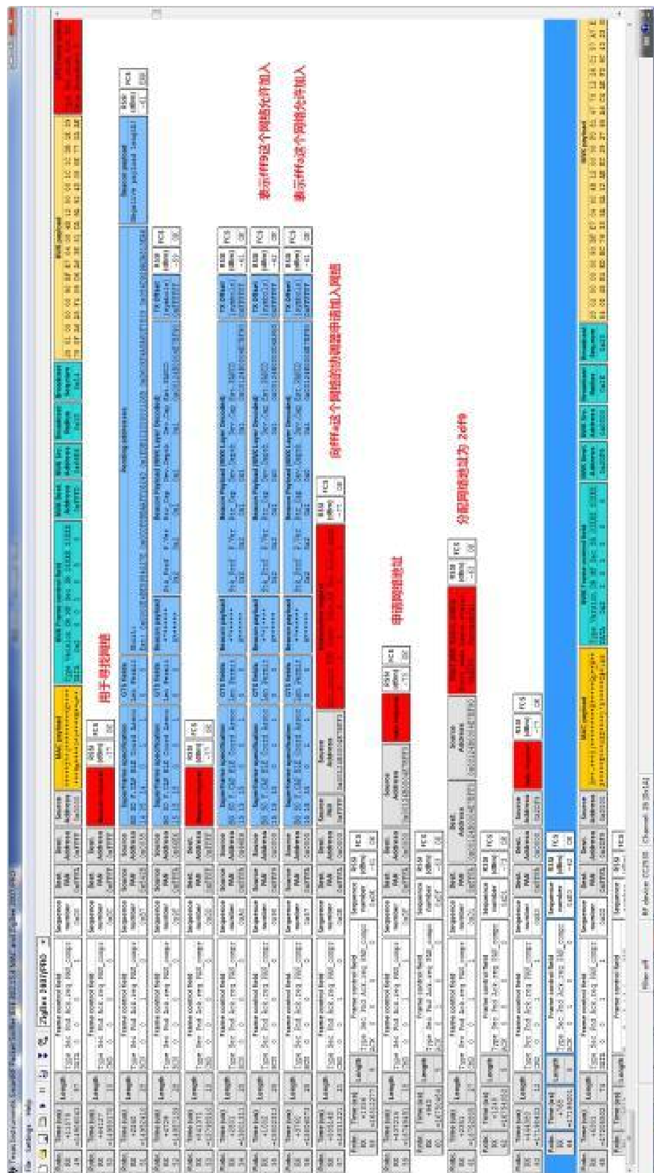


图6-5　新节点入网过程的通信数据

6.1.6　ZigBee的应用层

应用层（Application Layer）是ZigBee标准中定义的最高层，定义了ZigBee应用对象（Application Object）的操作接口，应用对象由ZigBee联盟或者ZigBee产品厂家定义，ZigBee联盟定义了很多标准的常用的应用对象。

ZigBee设备对象（ZigBee Device Object，ZDO）是每个ZigBee设备都必须实现的，ZDO实现的功能有设置设备角色（比如协调器、路由器或者终端节点）、安全服务（比如设置或删除密钥）、网络管理（比如节点关联、去关联），ZDO定义了一个ZigBee设备Profile（ZigBee Device Profile，ZDP，会在后面讲解），ZDP绑定在编号为0的应用终结点（Endpoint）上。

6.1.7 ZigBee的应用支持子层

应用支持子层（Application Support Sublayer，APS）通过一套通用的服务为ZigBee应用层和网络层之间提供了接口，ZDO和厂家自定义的应用对象可以使用这些服务。APS通过数据实体（APSDE）和管理实体（APSME）来提供这些服务。APSDE通过APSDE-SAP来提供数据传输服务，APSME通过APSME-SAP来提供管理服务，并维护一个叫作APS信息库（APS Information Base，AIB）的被管理对象数据库，如图6-6所示。

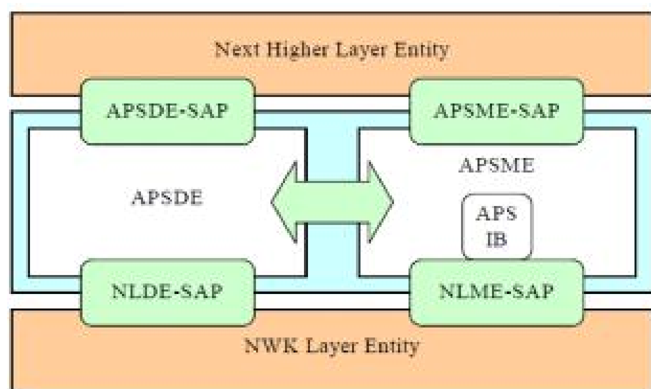


图6-6 应用支持子层的结构

应用支持子层数据实体（APSDE）通过其对应的服务接入点APSDE-SAP提供的服务如下：

- 生成应用层数据包—将应用层数据包（PDU）加上协议头，从而生成应用支持子层数据包。

- 绑定—一旦两个设备被绑定，这两个设备就可以进行通信了。

- 组播地址过滤—可以根据应用终结点的组属性来过滤以组播地址为目的地址的数据包。

- 可靠数据传输—通过端到端的重传机制，在网络层的基础上进一步提高数据传送输的可靠性。

- 重复数据过滤。

- 分包—如果一个消息超过了网络层所能传送的最大包长度，就可以对数据包进行分段传送，同时在接收到被分段传送的数据时也由APSDE进行重新组装。

应用支持子层管理实体（APSME）为应用于协议栈之间提供了交互接口，通过其对应的服务接入点APSME-SAP提供的服务如下：

- 绑定管理—提供通过两个设备服务和需求来匹配两个设备的功能。

- AIB管理—提供读取及修改AIB内属性的功能。

- 安全—提供通过密钥来认证，以及与其他设备建立联系的功能。

- 组管理—提供通过一个地址来寻址多个设备（组播寻址），以及往组里添加或从组里删除设备的能力。

6.1.8 ZigBee应用Profile

应用Profile定义了消息、消息格式，以及对这些消息的处理过程，在不同的设备上运行的应用实体如果遵循相同的Profile就可以实现交互，例如常用的Profile有智能家居，不同厂家生产的设备只要遵循相同的Profile就可以实现交互。

6.2 ZigBee安全

ZigBee采用128位的AES加密算法，其安全架构决定安全性取决于对称密钥的安全保存、所使用的保护机制、加密机制及相关策略的正确实现等，所以信任安全架构的安全性实质上是信任密钥预置及初始化和密钥的使用过程及保存。

ZigBee采用CCM*来提供数据完整性和加密，CCM*注 是基于CCM（Counter mode with Cipher Block Chaining Message Authenticity Check）改进而来的算法，可以提供只加密、只完整性认证以及同时加密和完整性认证三种保护方式。如图6-7所示为在ZigBee中CCM*被用于数据加密及数据完整性认证。图中的MIC（Message Integrity Check）是指消息完整性认证。

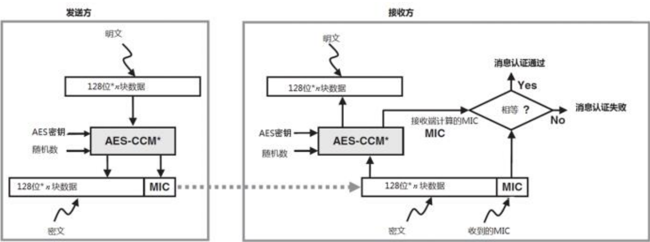


图6-7　在ZigBee中CCM*被用于数据加密及完整性认证

1. 关于CCM*可以参考https：//en.wikipedia.org/wiki/CBC-MAC。返回

6.2.1 安全层次

ZigBee设备实际上可以在三个逻辑层实现安全，这三个层分别是MAC层、网络层、应用支持层，它们都采用128位的AES加密算法，如图6-8所示。

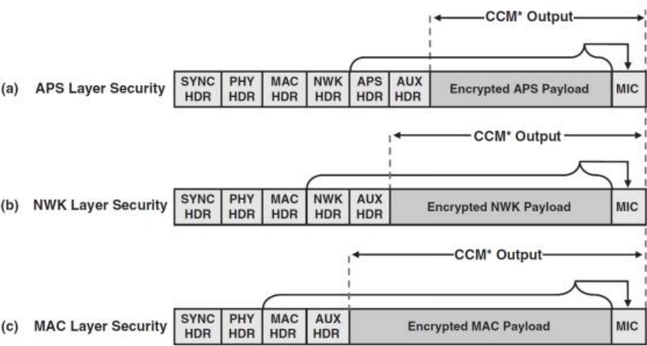


图6-8　ZigBee设备在三个逻辑层加入安全机制㊟

但由于MAC是由IEEE802.15.4标准定义的（而且一般没有使用），所以对于ZigBee安全我们只讨论网络层和应用支持层安全机制。而且每层的安全都是独立的，互不关联。

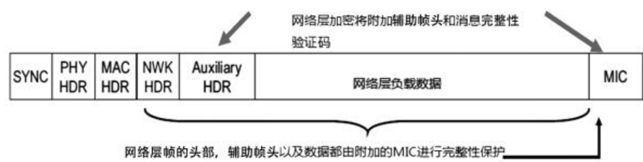


图6-9　使用了网络层安全机制的数据帧

使用了网络层安全机制的数据帧如图6-9所示。当网络层采用安全机制时会引入辅助帧头（Auxiliary HDR）和MIC。

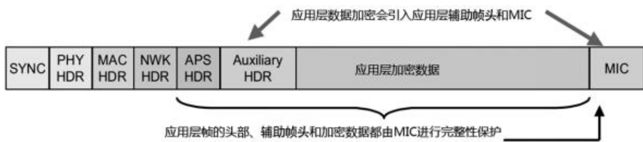


图6-10　使用了应用层安全机制的数据帧

与网络层类似，使用了应用层安全机制的数据帧如图6-10所示。当应用层采用安全机制时会引入辅助帧头（Auxiliary HDR）和MIC。注意，这里的辅助帧头和MIC与网络层的辅助帧头和MIC是相互独立的。

1. 此图引用自ZigBee wireless networks and transceivers，作者Shahin Farahani。返回

6.2.2 密钥类型

ZigBee定义了三种密钥类型：主密钥、网络密钥、链路密钥。

- 主密钥（Master Key）：这种密钥是部署设备时就已经存储在设备里面的，主要用于在进一步的密钥协商过程（Symmetric Key Key Establishment，SKKE）中对链路密钥进行保护。

- 网络密钥（Network Key）：128位的网络密钥是由网络中所有的节点共享的，用于加密组播和广播数据。网络密钥有可能是在节点入网时明文发送的。

- 链路密钥（Link Key）：这是每两个节点间共享的密钥，用于对两个节点间的通信进行加密，主要由应用层来管理，但在实践中这种密钥基本没有使用。

6.2.3 安全等级

1.高级安全/商用模式（Commercial Mode）

信任中心（一个网络只有一个，一般由协调器担任）与网络上所有设备共享主密钥和链路密钥，这种模式对信任中心的存储资源要求较高，如图6-11所示。

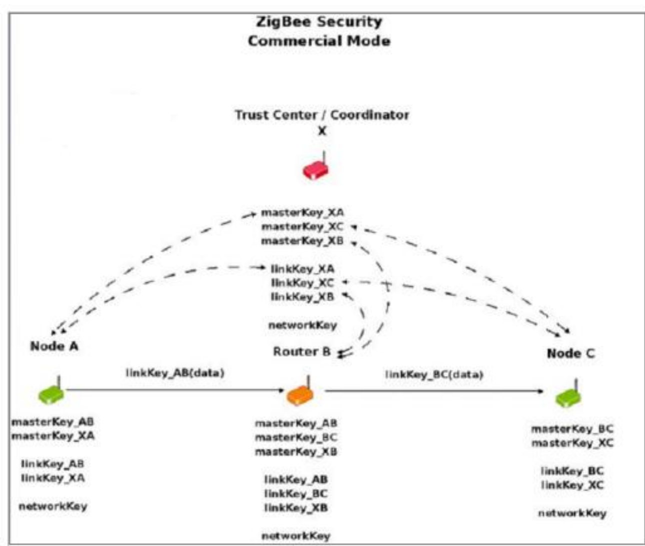


图6-11　高级安全/商用模式示意图

2.标准安全/家用模式（Residential Mode）

信任中心只与所有节点共享网络密钥，因此对资源要求较低，适合嵌入式应用，这是很多传感器网络所采用的模式，如图6-12所示。

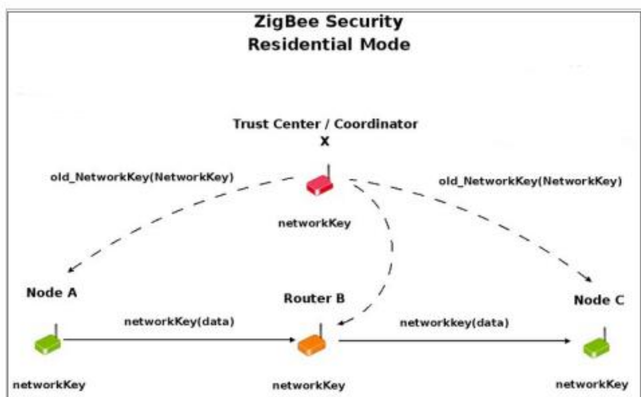


图6-12　标准安全/家用模式示意图注

1. 此图片来源于http://www.libelium.com/security-802-15-4-zigbee/#!prettyPhoto。返回

6.2.4 密钥分发

密钥分发、更新、废除对于ZigBee网络安全部署非常重要，部署人员可以使用对称密钥交换协议（Symmetric Key Key Establishment，SKKE）来派生网络密钥和链路密钥（前提是要求每个节点都已经有一个主密钥）。另外，还有两种密钥分发方式，即密钥传输和密钥预置。

- 在密钥传输方式中，网络密钥和链路密钥被明文传送，攻击者可以通过抓包来获取密钥，从而解密后续通信数据或者伪装为合法节点。

- 在密钥预置方式中，部署人员将在生产设备时把密钥预先存储到ZigBee节点中。这种方式虽然安全，但是不便于密钥的更新和废除，当网络结构发生变化时需要手动对网络节点进行重新配置。

6.2.5　ZigBee节点入网认证

对入网的节点进行认证有如下三种方式。

- ACL（Access Control List）：网络中的节点通过与之通信的节点的MAC地址（物理地址）来认证对方，每个节点都维护一个可以与之通信的节点的MAC地址列。这种方式只有与CCM*的数据认证功能（数据加密功能是可选的）相结合时才能发挥作用，因为如果没有使用CCM*对消息进行认证，则攻击者可以对MAC地址进行伪造。

- 在标准安全/家用模式中，一个节点在入网通信之前必须由信任中心发送网络密钥对其进行授权（是否对其授权可以由MAC地址来判断），当网络密钥被预先部署在节点中时，信任中心向其发送全为0的密钥表示授权；如果网络密钥没有被预先部署在节点中，则信任中心以明文形式将密钥传送给加入节点（注意，这个过程是很危险的）。可以看出，在这种安全模式中，入网节点没有对信任中心进行认证而无条件接受了信任中心发送过来的密钥，如果节点没有预先设置网络密钥，则攻击者可以伪造信任中心来与其通信。

- 在高级安全/商用模式中，不允许对网络密钥进行明文传输，当节点想加入网络时需要使用对称密钥交换协议（SKKE）来生成（注意是生成）网络密钥，如果节点没有主密钥，则信任中心可以将密钥以明文形式发送给节点。在这里SKKE就不详细讲了，只需要知道其功能就是通信双方在已经有一个共享密钥的情况下对对方进行认证（零知识证明）并生成会话密钥。

6.3 ZigBee攻击

本章前面讲了很多基础知识，下面介绍针对ZigBee的攻击方法，以及使用到的工具软件和硬件。

6.3.1　攻击工具介绍

1.CC2531USB Dongle

如图6-13所示，这是德州仪器公司基于CC2531开发的、相当于一个带USB功能的ZigBee全功能设备，可以通过编程使其作为ZigBee网络中的任意节点，其自带的固件可以实现抓包功能，很多协议分析软件都与其兼容。

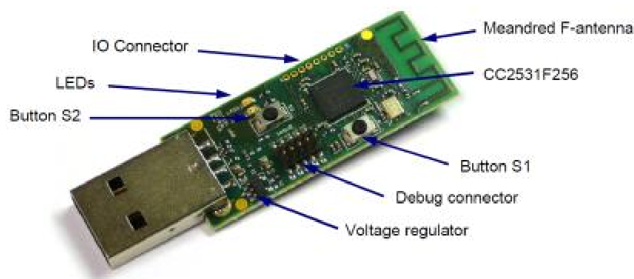


图6-13　CC2531 USB Dongle

2.CC Debugger

如图6-14所示，这是针对德州仪器公司的低功耗射频SoC芯片的烧写及调试器，可以与IAR嵌入式开发环境仪器配合使用来对ZigBee芯片进行调试及固件烧写，也可以与SmartRF Flash Programmer配套使用来读取设备固件。



图6-14　CC Debugger

3.Unicorn_ZigBee

如图6-15所示，这是360独角兽团队基于TI的CC2530芯片开发的ZigBee模块，可以用于重放数据、节点伪造等。



图6-15　Unicorn-ZigBee

4.IAR集成开发环境

IAR集成开发环境如图6-16所示，用于ZigBee应用程序开发，在

对目标系统分析完成后，可以使用IAR基于Unicorn ZigBee开发出攻击应用。

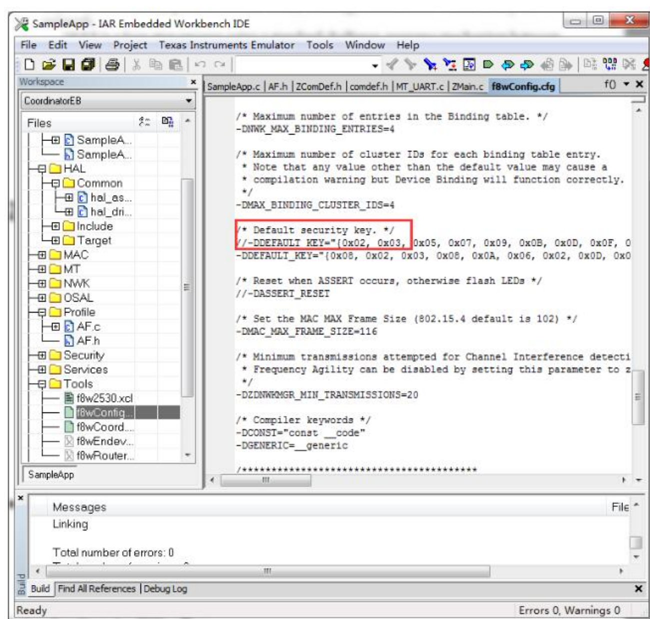


图6-16　IAR集成开发环境

6.3.2　协议分析软件

常用的协议分析软件有Ubiqua Protocol Analyzer、Perytons Protocol Analyzer、TI Packet Sniffer等，它们都兼容CC2531USB Dongle。

1.Ubiqua Protocol Analyzer

如图6-17所示，这是由Ubilogix开发的，Ubiqua集成了基于IEEE802.15.4的协议解析器和分析器，具有友好的用户界面，非常容易上手。

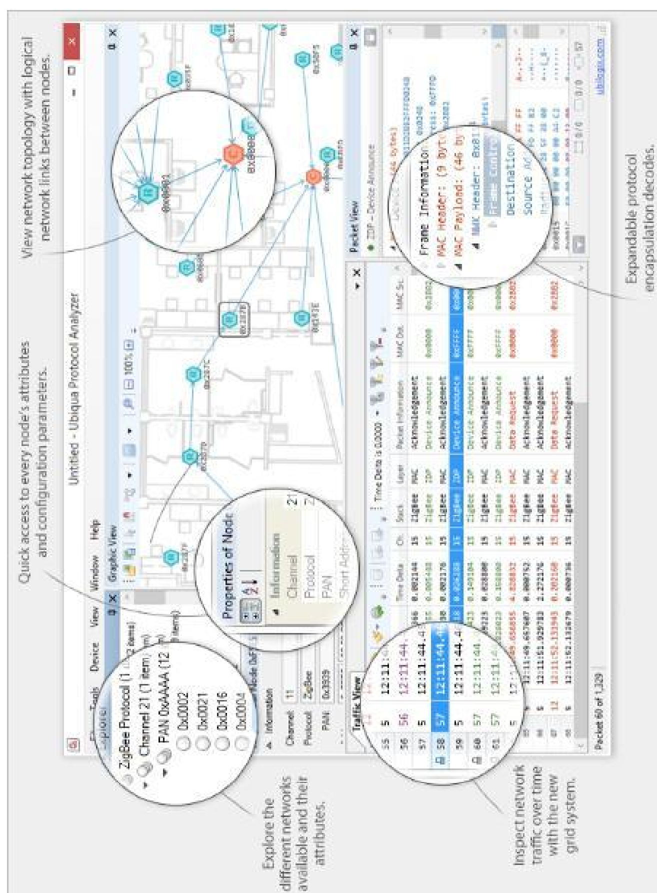


图6-17　Ubiqua Protocol Analyzer

2.Perytons Protocol Analyzer

如图6-18所示，这是与Ubiqua类似的协议分析软件，可以用于分析IEEE802.15.4、ZigBee、6LoWPAN、RF4CE、Thread、Bluetooth Smart、PLC-Prime、G3-PLC，以及其他无线或者有线通信协议。

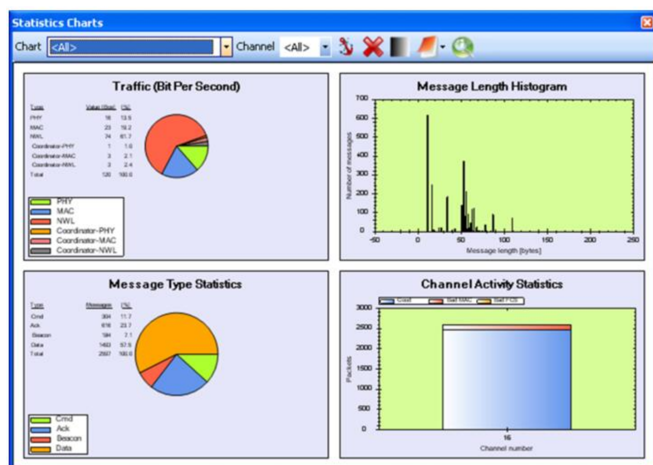


图6-18　Perytons Protocol Analyzer㊟

3.Packet Sniffer

如图6-19所示，这是德州仪器公司官方推出的抓包工具，支持多种无线通信协议，如ZigBee、BLE等。

Ubiqua和Perytons都具有对数据进行解密的能力，如图6-20所示。

如果能找到可能的密钥，把这些密钥输入到密钥列表里，Ubiqua就可以依次尝试直到解密成功。

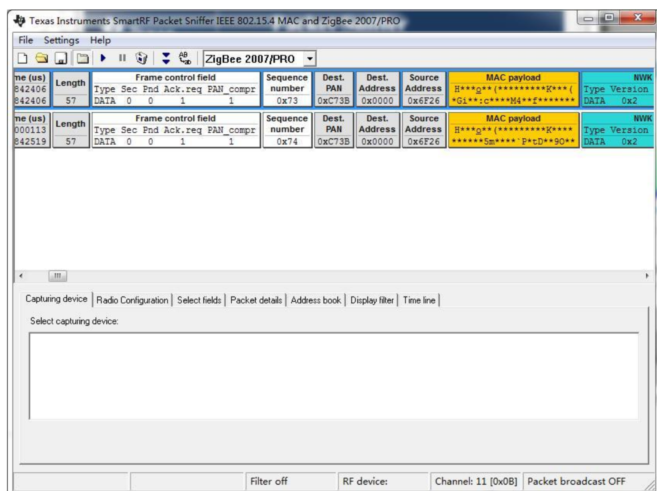


图6-19 Packet Sniffer

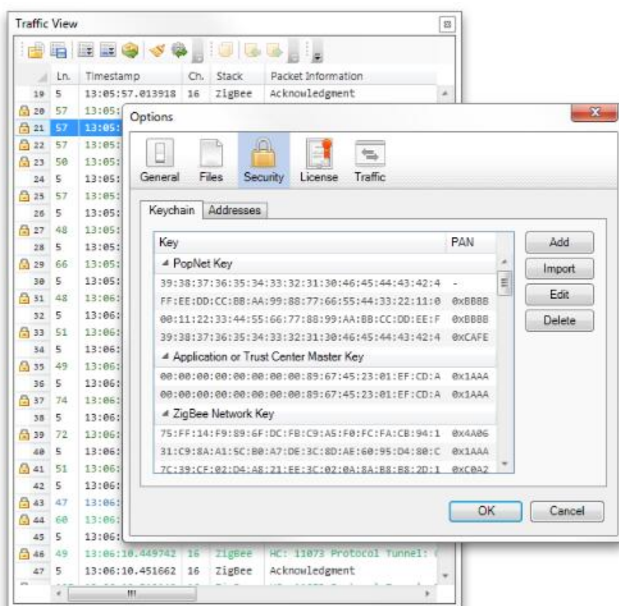


图6-20 解密数据

4.KillerBee

KillerBee是一套基于Python的ZigBee攻击框架，可以从http://killerbee.googlecode.com下载，它是由Joshua Wright在Linux系统下开发的，整个框架包括了多种针对ZigBee的攻击工具，如用于重放攻击的zbreplay、用于网络发现的zbstumbler、用于网络窃听的zbdump等。要使用KillerBee框架，需要用到RZUSBSTICK，如图6-21所示。



图6-21　RZUSBSTICK

RZUSBSTICK是由Atmel公司推出的支持2.4GHz频段的IEEE802.15.4标准的设备，Atmel开源了RZUSBSTICK的固件源码，所以开发者可对其固件进行修改来适应自己的应用。值得一提的是，KillerBee在使用RZUSBSTICK时对其固件进行了修改，所以在使用KillerBee框架之前必须重写RZUSBSTICK的固件，具体操作可以查看KillerBee的使用文档。

1. http://www.perytons.com/index.php/protocol-analyzers/zigbee/返回

6.3.3　网络发现

网络发现分为被动发现和主动发现，被动发现是指通过监听网络的数据通信来发现网络；主动发现是指利用ZigBee的信标请求（Beacon Request）来主动探测网络。

1.被动发现

利用CC2531USB Dongle与上面所介绍的任意一种协议分析软件就可以进行网络发现，只需要在某个信道上进行监听就可以发现这些网络中通信的节点，如图6-22所示。

从图6-22中可以看出，在信道11上有一个网络，其网络号为0x2c86，网络上有7个节点，其中协调器的网络地址为0x0000。被动发现的缺点是速度较慢，只有在网络中有数据通信时才能发现网络。

2.主动发现

主动发现是通过发送信标请求（Beacon Request）帧来实现的，当信道上的路由器或协调器收到信标请求帧时会回复信标，如图6-23所示。

从图6-23中可以看出，通过信标就可以找到这个网络号（PAN ID）为0x8080的网络。通过信标还可以知道网络的其他信息，如图6-24所示。

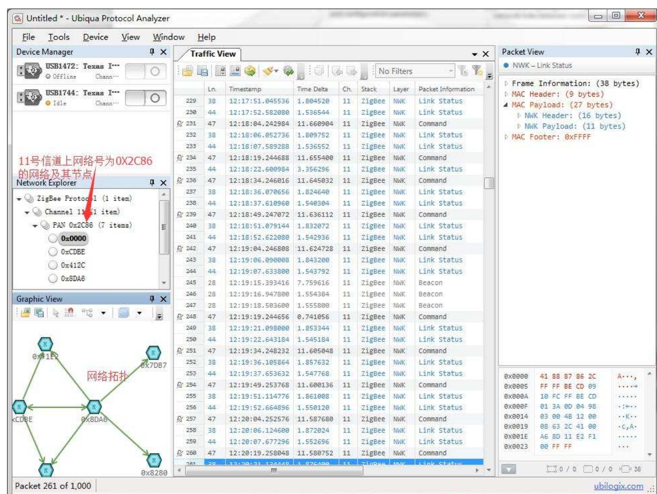


图6-22　被动网络发现

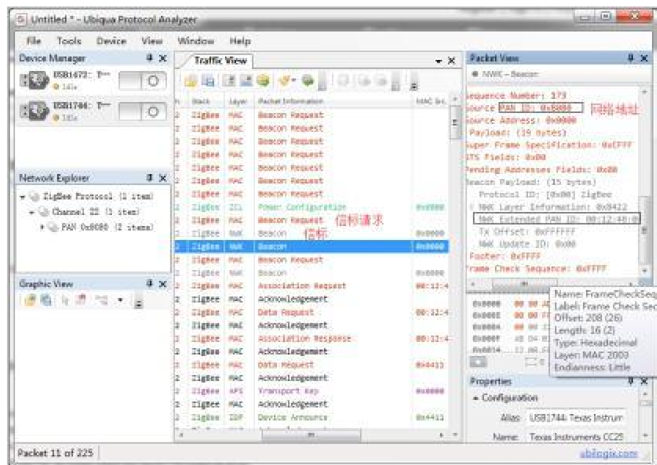


图6-23　主动网络发现

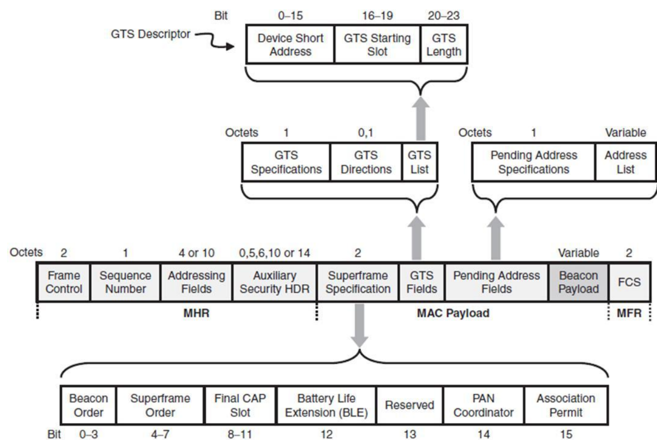

GTS Descriptor
Bit
0–15
16–19
20–23
Device Short Address
GTS Starting Slot
GTS Length
Octets
1
0,1
GTS Specifications
GTS Directions
GTS List
Octets
1
Variable
Pending Address Specifications
Address List
Octets
2
1
4 or 10
0,5,6,10 or 14
2
Variable
2
Frame Control
Sequence Number
Addressing Fields
Auxiliary Security HDR
Superframe Specification
GTS Fields
Pending Address Fields
Beacon Payload
FCS
MHR
MAC Payload
MFR
Beacon Order
Superframe Order
Final CAP Slot
Battery Life Extension (BLE)
Reserved
PAN Coordinator
Association Permit
Bit
0–3
4–7
8–11
12
13
14
15

图6-24　信标帧结构

6.3.4 对非加密信息的攻击

对于没有加密的网络，可以对其进行窃听攻击，即使网络进行了加密，攻击者也可以对没有加密的数据加以利用，来获得网络配置信息、拓扑、节点地址，甚至分发的密钥。可以通过CC2531USB Dongle与上面提到的任意一种协议分析工具配合来实现网络窃听。下面以Ubiqua为例，如图6-25所示。

从图6-25中可以看出，即使网络进行了加密，我们也以获得一些有用的信息，如网络拓扑结构、信道上运行的网络等。值得一提的是，在左上角的设备管理窗口中可以选择当前可用的抓包设备（就是有几个类似于CC2531USB Dongle的设备插在计算机上），图中显示有两个设备，每个设备都可以设置为监听不同的信道。这些设备可以同时运行，也就是说，我们可以使用多个设备来同时监听运行在不同信道上的网络。来自不同信道的数据也会在数据窗口的Ch.字段中标示出来。

1.窃听攻击的防御

由于无线电通信的广播特性，数据在空间传播难免被攻击者窃听，攻击者的攻击目的无非就是为了获得通信数据或者伪造数据。在ZigBee中可以使用CCM*加密机制提供的加密和消息认证码来防止数据被理解或者被伪造篡改，设置复杂的密钥，以及对密钥的安全保存和处理。

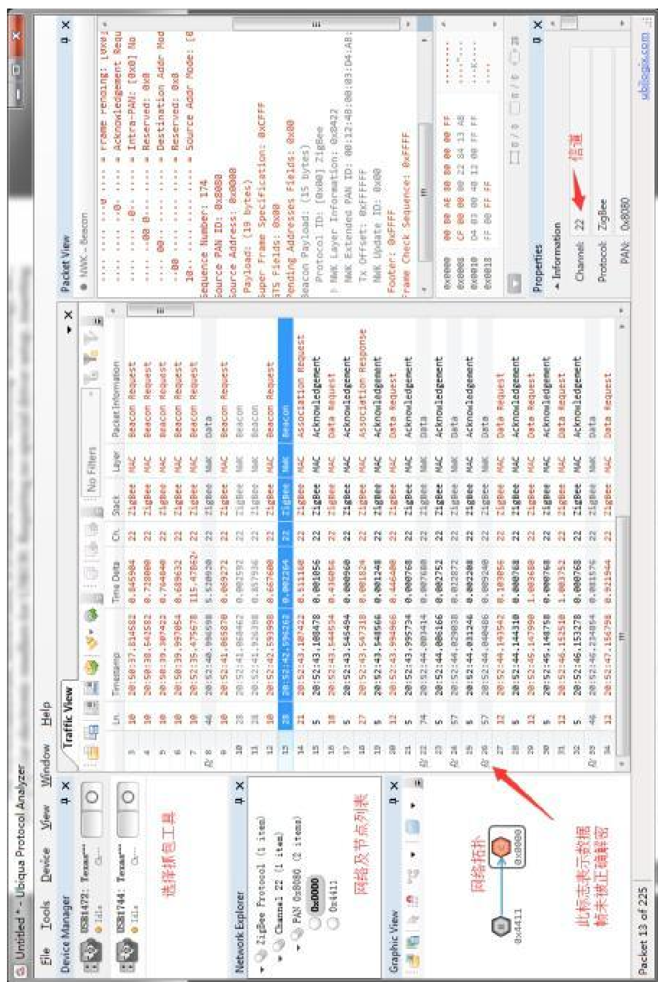


图6-25　ZigBee窃听实例

2.重放攻击

重放攻击的原理是重新发送所抓取到的数据。这里提到的重新发送是广义的，包括但不限于对物理层整个数据帧进行重放，因为ZigBee数据分为多层，重放可以在应用层、网络层、MAC层等实现，某些层会加入帧序列号来防止重放或者防止重复数据，更改帧序列号再发送也属于重放攻击，这些都要视具体情况而定。重放攻击的效果取决于被重放数据的作用，例如，在智能家居应用中攻击者可以通过

重放开关煤气阀的信号来开启煤气阀。

根据已经公布的漏洞信息显示，有的ZigBee协议栈实现可以被重放攻击，例如德州仪器公司的灯光开关应用，攻击者在用户正常使用开关控制灯光时对数据进行窃听，然后就可以重放这些控制数据（如开关灯的数据）来控制用户家的灯光。还可以将这种重放攻击与其他方式结合，比如在小偷闯入时将灯光关闭从而逃脱摄像头的监控，也可以是简单快速地开关等来和用户开玩笑。

重放攻击可以通过多种方式实现，例如软件无线电（SDR）、ZigBee模块、KillerBee框架提供的ZBreplay，以及其他ZigBee调试器。假如图6-26所示是一个控制阀门的数据包，其功能是开启阀门。

由于没有采取加密措施，攻击者可以对其数据进行重放，从而控制阀门。

3.重放攻击的防御

为了防止重放攻击，开发者在开发ZigBee应用时应该配置协议栈使其对收到的数据的序列号进行检查。而且，由于网络层序列号仅有一个字节，所以只有256个不同的序列号。因此，如果仅仅靠网络层序列号来检查数据是不够的，因为在一个周期（256个数据帧）之后该数据帧会通过数据帧检查而被认为是合法的数据帧，所以必须在应用层加入序列号来辅助校验。

6.3.5　对加密信息的攻击

在ZigBee网络中，由于部署场景、人机接口、设备资源等的限制使得密钥分发、更换、注销和管理非常具有挑战性。在其使用场景中（例如智能家居）往往没有专业的管理员对网络进行维护，用户不懂技术，因此网络在部署以后密钥一般就固定不再改变了。而且，由于很多厂家生产的产品要保证兼容性（比如用户想在已有的智能灯光系统中加入一个灯泡）常常给一个系列的产品设置相同的密钥，或者采用密钥分发的方式在新设备入网时由协调器将网络密钥发送给新设备，这些都给攻击者以可乘之机。

1.密钥分发攻击

密钥分发攻击主要是指在新节点入网向信任中心请求密钥时对密钥进行窃听，有多种工具都可以用于密钥窃听，比如Perytons、TI Packet Sniffer、Ubiqua、KillerBee的zbdsniff等。如图6-27所示是通过Packet Sniffer抓取到的密钥分发过程。从图中可以看出，虽然在新设备收到密钥后通信立即被加密了，但是密钥已经暴露。

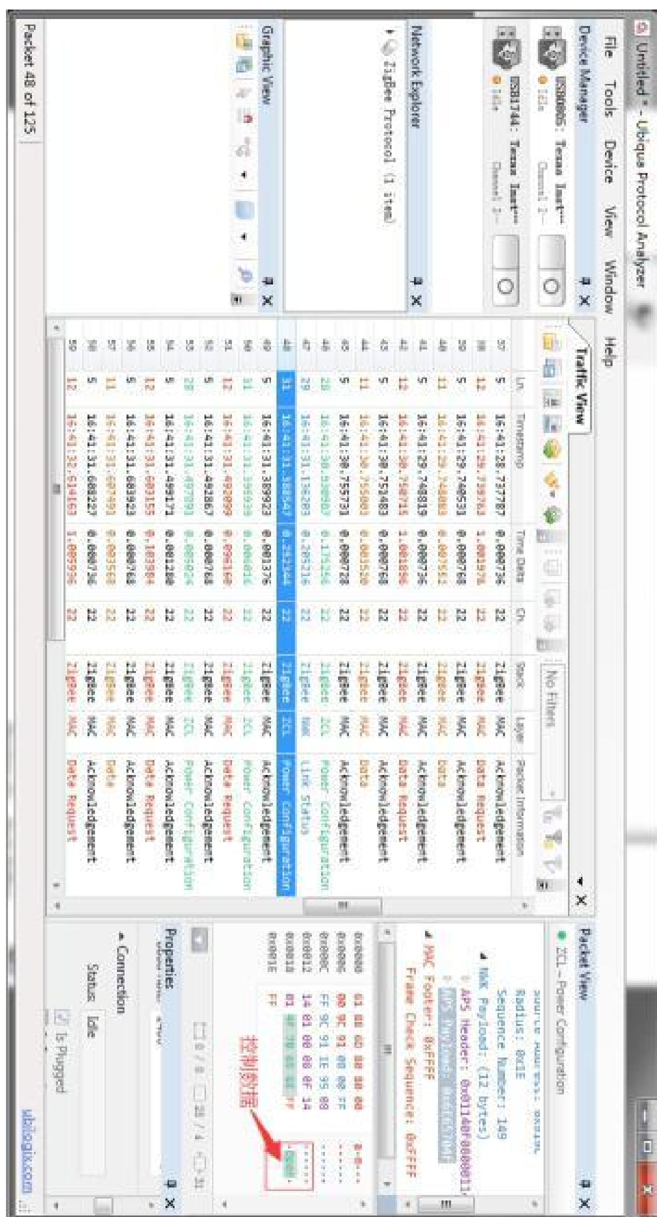


图6-26　控制阀门的数据包

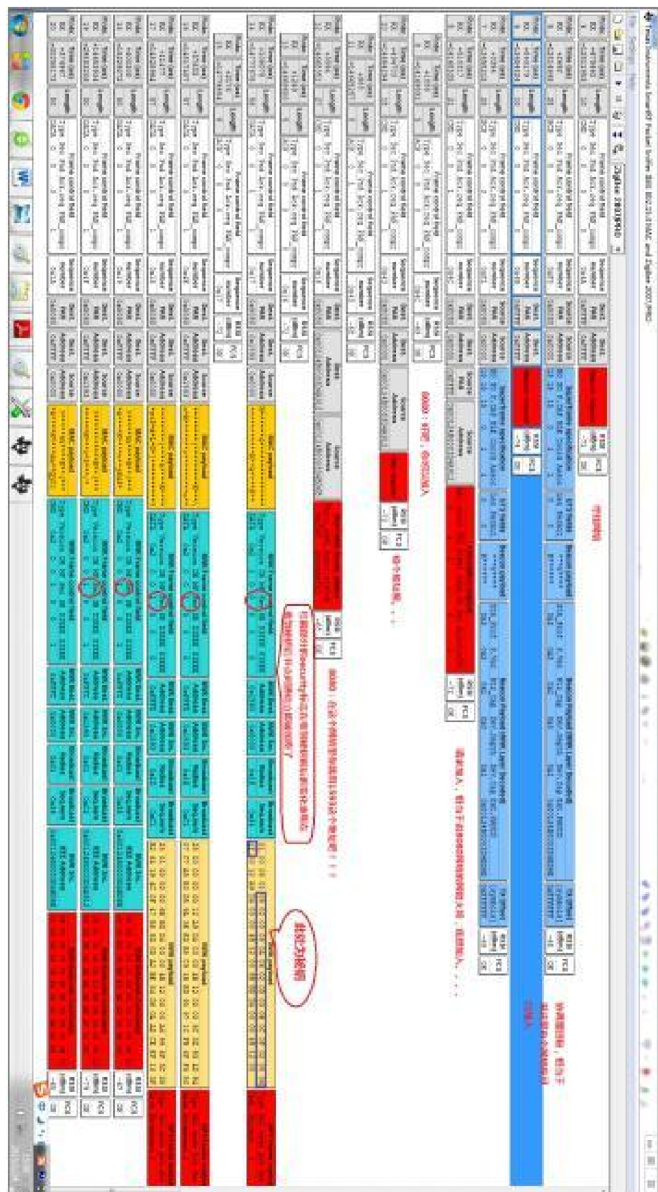


图6-27　通过Packet Sniffer抓取到的密钥分发过程

Packet Sniffer不提供解密功能，但是密钥已经暴露，攻击者可以利用密钥对通信解密，或者利用ZigBee模块开发应用时将这个密钥预置到应用中来解密后续通信。

Ubiqua协议分析仪也是一个很易于使用的工具，当使用Ubiqua来侦听上述密钥传送过程时，Ubiqua可以在收到密钥传送帧后自动提取并保存密钥并且解密后续数据。图6-28和图6-29显示的是通过Ubiqua监听到的被传送的密钥并且自动解密数据。

图6-28　密钥传送

图6-29　Ubiqua自动提取密钥并解密数据

2.密钥分发攻击的防护

为了防止密钥在传送过程中被窃取，可以使用其他密钥部署方式，比如密钥预置、密钥派生。密钥预置是指将密钥预置在设备中，避免在设备部署后通过无线来传送密钥。这种方式比较安全，但是不利于密钥的更新。密钥派生是指利用SKKE机制派生出链路密钥，但是在进行密钥协商之前必须在设备中预置主密钥。

6.4 攻击实例

本节内容是在DEFCON23会议上分享过的，“I’m ANewbie Yet ICan Hack ZigBee–Take Unauthorized Control Over ZigBee Devices”。

6.4.1　从设备中获取密钥

正如前面所提到的，如果采用密钥预置或者密钥派生的密钥部署方式，那么预置的网络密钥或主密钥需要与固件一起存储在设备的ROM中，而这些密钥是网络中所有设备所共享的，如果我们能够读出设备的固件就可以设法找到密钥。下面以一个智能灯泡为例来演示如何从设备中提取密钥，然后解密数据，最后完全控制目标设备。目标设备的系统架构图如图6-30所示。

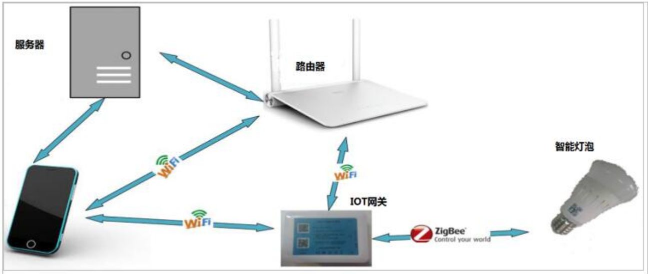


图6-30　智能灯泡系统架构图

在正常情况下，用户通过手机来控制灯泡，网关起协议转换作用，将手机通过WiFi传来的控制命令提取出来再通过ZigBee协议发送给灯泡。手机的控制数据可能通过以下几条链路到达灯泡，如图6-31所示。

攻击的目的是通过破解IOT网关与灯泡之间的加密信道，然后用攻击者自己的设备来伪造IOT网关来控制灯泡，如图6-32所示。

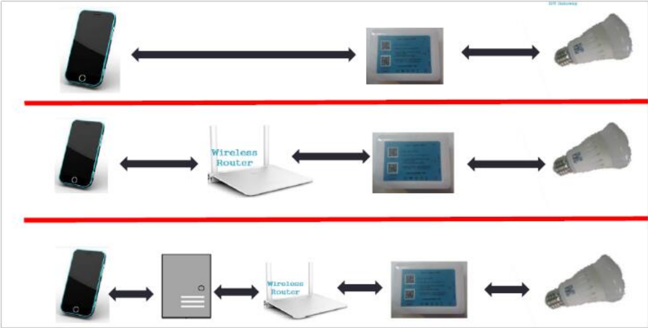


图6-31　手机的控制数据到达灯泡的几条链路

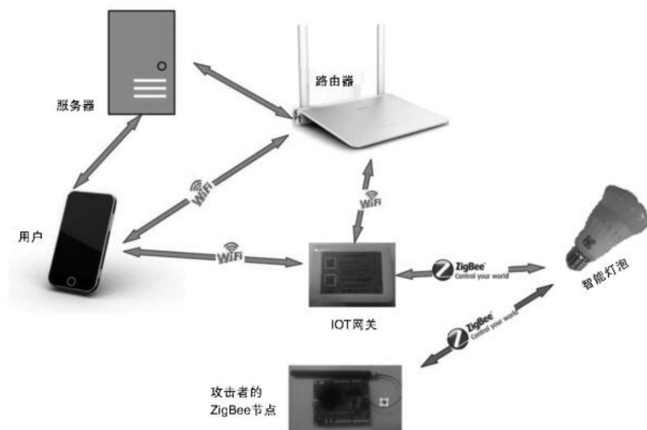


图6-32　针对目标系统的攻击场景图

通过前面的方法对网络进行窃听，发现数据是通过预置的网络密钥来加密的，也就是前面讲的标准安全/家用模式，如图6-33所示。

网络密钥在网络上的每个节点中都有，在这里是指网关和所有灯泡，如图6-34所示。

由于网关比灯泡更容易撬开，所以决定从网关下手，如图6-35所示为被撬开的网关。

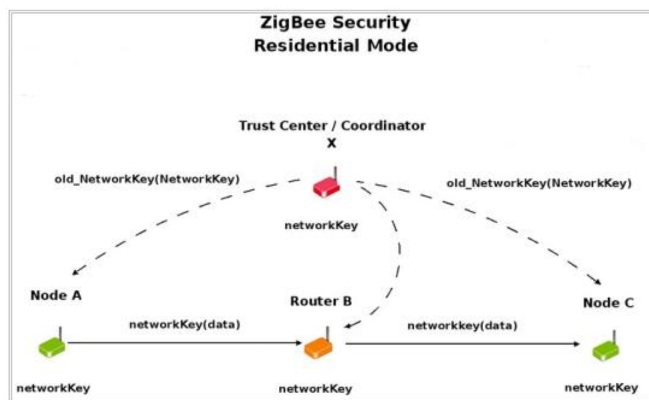


图6-33　ZigBee安全家用模式示意图



图6-34　灯泡和网关



图6-35　被撬开的网关

将CC Debugger与网关的调试接口相连接，然后使用TI的SmartRF Flash Programmer将固件读出来，如图6-36所示。

将固件读出来之后，要从固件中找到密钥有多种方法，如反汇编、遍历等。

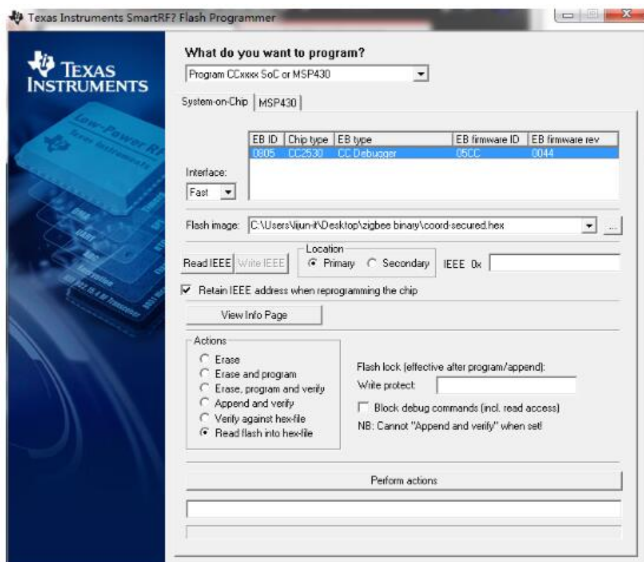


图6-36　使用TI的SmartRF Flash Programmer读出固件

KillerBee框架提供了一个工具ZBgoodfind专门用于在固件中寻找密钥，其主要原理是首先抓取加密数据包，然后尝试使用固件里所有连续的16字节来解密，如图6-37所示。

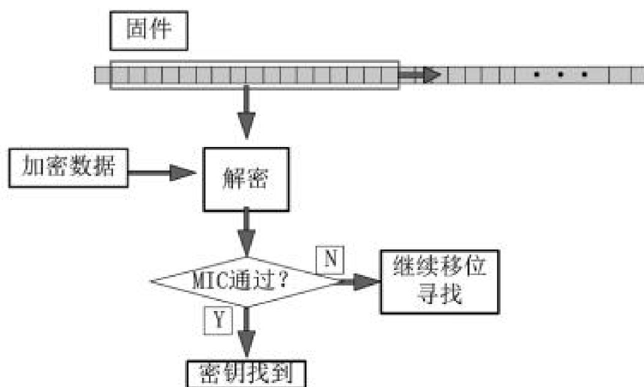


图6-37　ZBgoodfind的工作原理示意图

在图6-37中，将固件看做一个字节序列，尝试用所有连续的16字节来解密数据，直到解密成功。MIC（Message Integrity Check，消息完整性验证）用于验证解密是否成功，继续移位是以字节为单位进行移动的。图6-37也可以用伪码来表示：

```
for(index=0;index<sizeof(firmware)-16;index++)
{
    string key=firmware[index - （index+16）]；
    if(AESdecrypt（key，encryptedPacket）==SUCCESS)
    {
        printf("congratulations，key found！key=",key);
    }
    Break;
}
```

在固件中寻找密钥还有一种更快的方法，就是利用处理密钥的指令特征，由于很多产品都是基于芯片厂家的协议栈实现来开发应用的，而底层的密钥处理都由协议栈实现，所以密钥处理指令的特征在不同产品中可能都存在。下面以最常用的ZStack协议栈为例。

ZStack一般使用IAR来开发，在程序中有一个将此密钥转存到NV（Non-Volatile Memory）的动作，调用了memcpy函数，通过IAR调试跟踪到此语句后发现其有比较固定的模式，如图6-38所示。

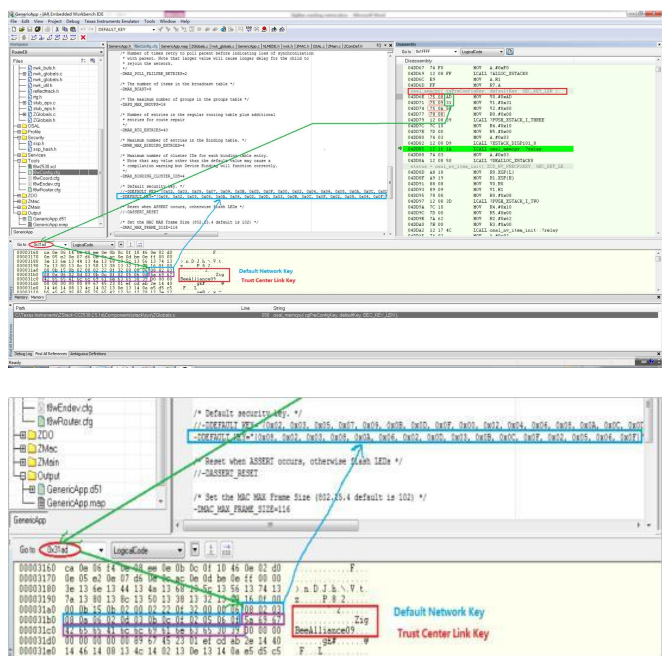


图6-38　通过操作密钥的指令来找到密钥地址

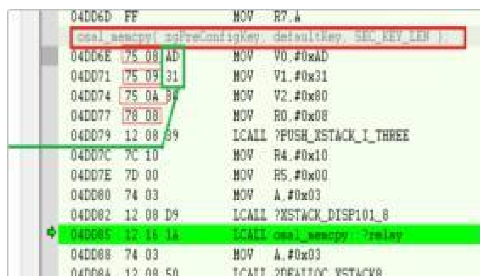


图6-38　通过操作密钥的指令来找到密钥地址（续）

图中标出的部分是比较固定的，75对应于mov指令的机器码，v0~v2对应的08~0a也是固定的，只有AD与31是不固定的，为密钥在存储器里的地址（0x31ad）。在0X31ad处存储了密钥“ZigBeeAlliance09”，通过查询标准文件知道这是默认密钥。

对于被攻击的节点，通过Hex编辑器使用此指纹或者叫作特征（例如在IDA中可以用7508？7509？750A）对其固件进行搜索，可以找出可能的密钥地址，如图6-39所示。

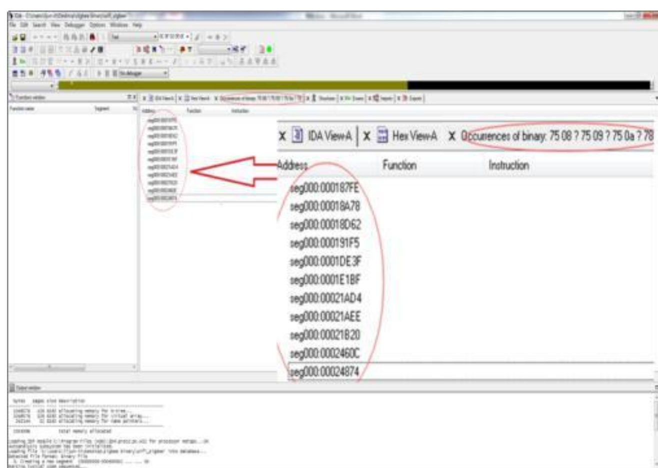


图6-39　操作密钥的指令机器码

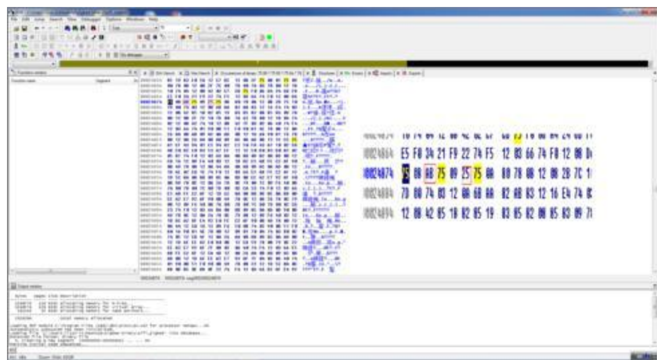


图6-39　操作密钥的指令机器码（续）

图中显示的密钥存储地址为0x25AB，找到0X25AB这个地址就可以看到AES密钥为080203080A06020D030B0C0F0205060F，共16B＝128bit，如图6-40所示。

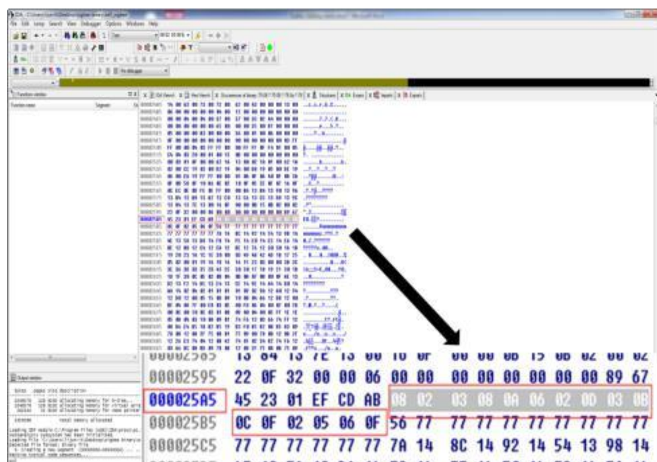


图6-40　密钥在flash中的位置

通过验证知道此为真实的网络密钥。

6.4.2　利用密钥可进行的攻击

- 密钥被破解之后可进行如下攻击：
- 应用层数据分析。
- 重放amp;伪造。
- 数据截取及篡改。
- 去关联攻击。
- 其他攻击。

1.应用层数据分析

数据解密成功后，要实现对目标系统的控制，我们还得对数据进行分析，分析结果如表6-2所示。

表6-2 数据分析结果

Byte0	0x04
Byte1	目标网络地址
Byte2	
Byte3	未知
Byte4	模式（各种预设的情景）
Byte5	红
Byte6	绿
Byte7	蓝
Byte8	亮度
Byte9	校验（前面所有 Byte 异或）

数据长10Byte，最后1字节为前面所有字节的异或校验，1~2字节为要控制的设备的PANID。有了这些信息，我们就可以用自己的节点来控制目标系统了。

2.重放amp;伪造

有了密钥之后，要对数据进行重放还需要对图6-41中标出的部分（各序列号也要更改）进行伪造。比如，若不对IEEE地址和网络地址

进行伪造，那么数据会被目标节点的源地址过滤机制（白名单在Associate组网时建立）过滤掉，不会传送到目标节点的应用层，从而无法达到控制目标节点的目的，所以要对这些数据进行匹配。

由于我们对目标系统的应用层数据已经分析清楚，没有必要进行简单的重放攻击，可以直接伪造节点来加入系统取得完全控制权，如图6-42所示。

我们利用ZigBee模块来伪造节点加入网络，如图6-43所示。

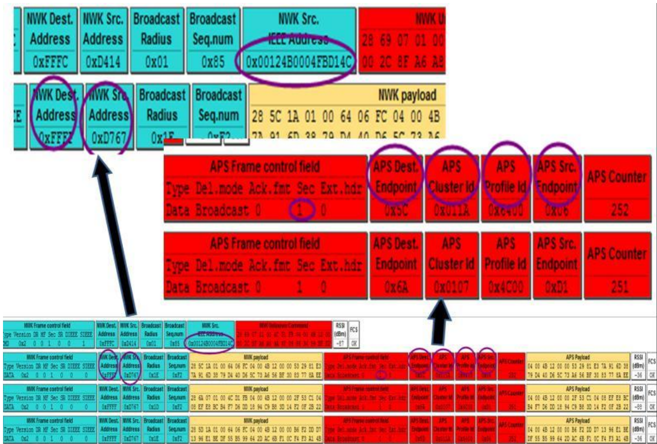


图6-41　数据伪造要点

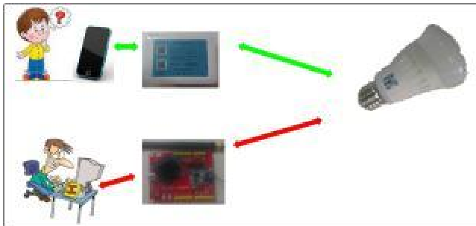


图6-42　直接伪造节点加入系统取得完全控制权示意图

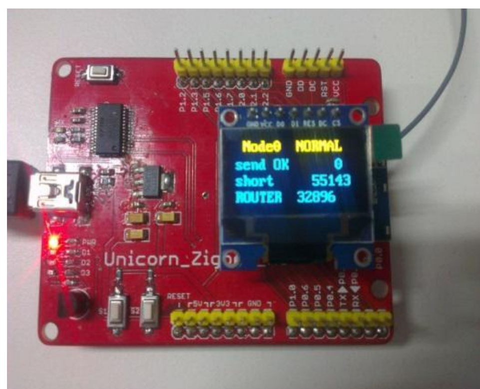


图6-43　ZigBee模块

值得注意的是，我们前面分析的数据是应用层数据，要对目标系统实现控制还需要对终结点（Endpoint）、簇ID（Cluster ID）等进行伪造，如图6-44所示。

3.数据截取及篡改

如本章开头所介绍的，由于在ZigBee网络中除协调器以外的节点大都为电池供电，为了降低功耗，这些节点大部分时间都工作在休眠状态，只是定期退出休眠模式来接收或发送数据。也就是说，当有节点试图向休眠节点发送数据时数据不能被立即接收，而先由协调器缓存，等待节点退出休眠模式后再发送给节点。如图6-45所示为ZigBee协调器向网络中其他节点发送数据的流程图。

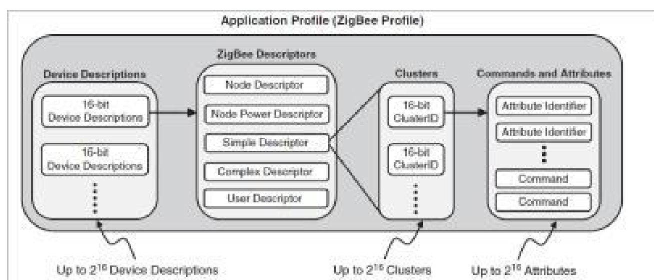


图6-44　对终结点、簇ID等进行伪造

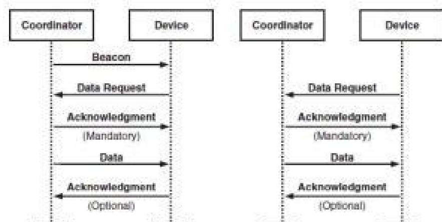


图6-45　ZigBee协调器向网络中其他节点发送数据的流程图注

从图6-45中可以看出，在使用信标的网络中，当协调器有要发送给某节点的数据（或者有为某节点缓存的来自其他节点的数据）时，协调器会在信标帧中标明有数据等待接收，对应节点收到信标后就向协调器请求数据。在未使用信标的网络中，则由数据的目标节点向协调器查询是否有自己的数据。

通过以上分析可以发现，如果其他节点伪造数据的目标节点向协调器请求数据，则协调器会在将数据发送给伪造节点之后清除所缓存的数据，这样就达到了数据截取攻击的目的。同样，如果伪造节点再假装协调器来回复其他节点的数据请求（回复的数据可以通过修改先前向合法协调器请求到的数据而得来），则可以达到篡改数据的目的。

4.去关联攻击

去关联攻击（disassociation attack）是指通过伪造节点的离开网络请求来破坏网络的连通性。

5.其他攻击

如图6-44所示：

- ZigBee联盟定义了一系列Profile（类似于一种功能描述，比如一个灯至少要有开关功能），如果节点使用了ZigBee联盟定义的Profile，那么其控制命令就与官方标准相同，密钥一旦破解就可以被完全控制。
- 但如果节点没有使用ZigBee联盟定义的Profile，则可以尝试

使用ZigBee联盟定义的Cluster（相当于命令，例如调色）及Attribute（相当于属性，例如调光的亮度）来控制节点。

- 如果还是不行，则可以对其进行fuzz攻击、穷举攻击等。

1. 此图引用自ZigBee wireless networks and transceivers，作者Shahin Farahani。返回

6.5 攻防分析

- 针对密钥破解，不要在固件中保存明文密钥；不要使用OTA模式来发送明文密钥；在产品发布前将调试接口关闭（比如烧断熔丝），防止固件被读出。

- 针对能量消耗型攻击，可以在网络部署完成后关闭Association功能及Beacon回复功能。

- 针对节点捕获定位，可以对节点发射功率进行随机调整来使定位更难。

- 针对重放攻击，可以在应用层数据中加入时间戳、序列号等在一定程度上防止重放攻击。

- 针对密钥破解后数据被分析，可以在应用层加入自定义的简单加密算法来给攻击者增加难度。

第7章　移动通信网络安全现状

本章将讨论通信距离更远、覆盖更广的一类系统—移动通信网络，包括2G，也就是GSM网络，还有3G/4G网络，将讨论WCDMA和LTE系统。

7.1　GSM系统安全现状

7.1.1　GSM/UMTS系统术语和基本概念的简介

在介绍蜂窝网络之前，我们需要知道GSM网络中有很多英文缩写，为了方便读者理解以及行文方便，我们将常见的术语写在前面，当读者读到后面内容记不清概念的时候，可以翻阅本节以便加深理解。如果你想直接进入正题，也可以跳过这一节。

移动交换中心（Mobile Switching Center，MSC），是GSM/CDMA的主要业务交付节点（Service Delivery Node），负责语音呼叫和短信，以及其他业务（例如电话会议、传真及电路交换数据）的路由。

MSC建立（set up）和拆除（release）端到端连接，在呼叫过程中，处理可移动性（mobility）和切换（hand-over）需求，并处理计费和实时预付费账户监视。

网关移动交换中心（Gateway MSC，GMSC），用于确定被叫签约用户目前正位于哪个拜访MSC。它也与PSTN进行接口，所有的移动到移动呼叫，以及PSTN到移动呼叫，都通过一个GMSC路由。这个术语仅在一个呼叫的上下文中有效，因为任何一个MSC都可能同时提供网关功能和拜访MSC功能。然而，一些制造商会设计专用的大容量MSC，不负责与任何BSS连接，这些MSC对于它们所处理的大部分呼叫都是GMSC。

拜访移动交换中心（Visited MSC，VMSC），是客户当前正位于的MSC。与该MSC相关联的VLR将拥有该签约用户的数据。

移动交换中心服务器（Mobile Switching Center Server，MSCS），是从3GPP第4发行版开始，对MSC概念进行重设计中的一部分。MSCS是MSC的一个软交换变种（soft-switch variant），它为漫游到其所服务区域的移动电话提供电路交换的主叫（calling）移动性管理和GSM业务。MSCS功能使得控制（信令）和用户平面能够分离（网元中的集合信道被称为媒体网关/MG），从而保证了网络中网

元有更好的布置。

MSC连接到如下的网元：

归属位置寄存器（Home Location Register，HLR），用于获取关于该SIM和MSISDN（即电话号码）的数据。

基站子系统（Base Station Subsystem，BSS），处理与2G和2.5G移动电话之间的无线电通信。

UMTS陆地无线电接入网（UMTS Terrestrial Radio Access Network，UTRAN），处理与3G移动电话之间的无线电通信。

拜访位置寄存器（Visitor Location Register，VLR），当签约用户位于其归属网络之外时，提供签约用户的信息。

归属位置寄存器（Home Location Register，HLR），是一个中心数据库，包含被授权使用GSM核心网的各个移动电话签约用户的详细信息。每个公共陆地移动网络（Public Land Mobile Network，PLMN）中都可以有多个逻辑的或物理的HLR，但是一个IMSI/MSISDN配对，在同一时间只能被关联到一个逻辑的HLR（可能横跨多个物理节点）上。

HLR存储移动电话运营商所发行的每个SIM卡的详细信息。每个SIM卡都有一个唯一的识别码，称为“IMSI”，它是每条HLR记录的主键。

还有一个关联到SIM卡的重要数据项就是MSISDN，它是移动电话发起和接收呼叫所使用的电话号码。主MSISDN是用于发起和接收语音呼叫和短信的号码，但是一个SIM卡有可能拥有第二个MSISDN和它关联，用于传真和数据呼叫。每个MSISDN也是HLR记录的一个主键。只要一个签约用户还是该移动电话运营商的用户，就会有HLR数据存储。

HLR与如下的网元相连：

- GMSC，用于处理入向呼叫（incoming calls）。
- VLR，用于处理来自移动电话的请求，以附着（attach）到网络上。
- SMSC，用于处理入向短信。

鉴权中心（Authentication Center，AuC），是一个对GSM核心网连接尝试（通常是在电话开机时进行的）的SIM卡进行鉴权的功能。一旦鉴权成功，HLR就被允许管理该SIM卡以及上面描述的服务。也将生成一个密钥（encryption key），它将随后被用于加密所有的移动电话和GSM核心网之间的无线通信（语音、短信等）。

在AuC之中和之外的安全性的正确实现，是运营商避免SIM卡克隆的关键部分。

AuC不直接参与鉴权过程，而是生成被称为“三元组”（tiplets）的数据，供MSC在这个过程中使用。该处理的安全性依赖于AuC和SIM卡之间的共享密钥（shared secret），称为“K_i ”。Ki在制造SIM卡的时候被秘密地烧制在其中，同时被安全地复制到AuC中。这个K_i 从不会被在AuC和SIM卡之间传输，但是会被关联到IMSI来产生一个挑战/响应（challenge/response）用于身份识别，以及一个被称为“K_c ”的密钥在空中通信（air communication）上使用。

当MSC向AuC询问一个特定IMSI的新的三元组集合时，AuC首先生成一个被称为“RAND”的随机数，该RAND随后被关联到K_i 来生成如下两个数字：

- K_i 和RAND被送入A3算法，而“签名响应”（Signed Response，SRES）被算出。

- K_i 和RAND被送入A8算法，而被称为K_c 的会话密钥被算出。

号码（RAND，SRES，K_c ）形成一个三元组，被发回给MSC。当一个特定的IMSI请求接入GSM核心网时，MSC将三元组的RAND部分发送给SIM卡。该SIM卡将这个号码以及K_i （烧制在SIM卡中的）送入适当的A3算法，而SRES被算出并发回给MSC。如果SRES匹配三元组中的SRES（如果是一个合法的SIM卡，就应该匹配），则该移动电话就被允许附着（attach），并继续使用GSM业务。

在成功鉴权之后，MSC向基站控制器（Base Station Controller，BSC）发送密钥K_c ，这样所有的通信都可以被加密和解密。当然，移动电话可以通过将鉴权期间所提供的相同的RAND和K_i 一同送入A8算法，来自己生成K_c 。

通常AuC和HLR一起布置，但这不是必需的。尽管这个过程对于大部分的日常使用是安全的，但不保证不被突破。

A3算法被用于加密GSM蜂窝通信。实践中，A3和A8算法基本上被一起实现（称为A3/A8，参见COMP128）。按照3GPP TS43.020（在Rel-4之前是03.20）中的定义，A3/A8算法在SIM卡和GSM网络AuC中被实现，用于对客户进行鉴权，并生成一个密钥来加密语音和数据流量。尽管有现成的实现A3和A8算法的例子，但它们的开发被认为是各个独立的GSM网络运营商的事情。

拜访位置寄存器（Visitor Location Register，VLR），是一个数据库，用于存储漫游到一个MSC所提供服务范围的签约用户的数据。网络中的每个主要基站都严格地仅由一个VLR来提供服务（在MSC池的情况下，一个BTS可能由多个MSC提供服务），因此一个签约用户不会同时出现在超过1个的VLR上。

存储在VLR中的数据要么接收自HLR，要么从MS（Mobile Station，移动台）采集。实践中，由于性能原因，大部分供应商都将VLR与VMSC集成在一起，即使没有这么做，VLR也被通过适当的接口紧密地与MSC联系在一起。不论何时，一旦MSC发现它的网络中出现了一个新的MS，除了在VLR中创建一条新的记录外，它还更新该移动签约用户的HLR，将该MS的新位置告知它。如果VLR数据被损坏，则可能会导致出现短信和呼叫业务的严重问题。

当一个签约用户在处于该VLR区域内期间变为不活跃（inactive）状态时，清除该签约用户的记录。当超过一定时间都一直不活跃时，VLR删除该签约用户的数据，并通知HLR（例如，当该电话被关机并弃置不用，或者当该签约用户被移动到一个没有信号覆盖的区域超过很长时间时）。

当一个签约用户明确地移动到其他VLR时，遵照HLR的要求，删除该签约用户的记录。

设备身份寄存器（Equipment Identity Register，EIR），是一个数据库，存储关于移动设备的身份相关信息，以防止来自被盗的、未授权的或有缺陷的移动台的呼叫。一些EIR同时有能力对手机请求尝试记录日志，并将其保存在一个日志文件中。EIR通常和HLR集成在一起。EIR保留一个要被本网络禁止或监视移动电话的列表（由IMEI来区分），这被设计为允许对被盗电话的追踪。理论上，关于所有被盗移动电话的所有数据都应当通过一个中心EIR被分发到全世界所有的EIR。但是，很明显，在一些国家没有运营这项业务。EIR数据不必要实时更新，这就意味着此功能不必像HLR功能那样达到那么高的分布式程度。

国际移动用户识别码（International Mobile Subscriber Identity，IMSI），是用于区分蜂窝网络中不同用户的、在所有蜂窝网络中不重复的识别码。手机将IMSI存储于一个64比特的字段发送给网络，IMSI可以用来在归属位置寄存器（HLR）或拜访位置寄存器（VLR）中查询用户的信息。为了避免被监听者识别并追踪特定的用户，在大部分情况下手机和网络之间的通信会使用随机产生的临时移动用户识别码（Temporary Mobile Subscriber Identity，TMSI）来代替IMSI。

只要一个移动网络的用户需要与其他移动网络互通，就必须使用IMSI。在GSM、UMTS和LTE网络中，IMSI来自于SIM卡；在CDMA2000网络中，则直接来自于手机或者RUIM。

IMSI由一串十进制数字组成，最大长度为15位。少于15位的例子少见，例如，南非MTN有一些仍在网络中使用的较旧的IMSI为14位数字。IMSI由移动国家代码（Mobile Country Code，MCC）、移动网络代码（Mobile Network Code，MNC）和移动用户识别代码（Mobile Subscriber Identification Number，MSIN）依次连接而成。MNC长度为3位，MCC长度为2位（欧洲标准）或3位（北美标准），由MNC的值决定[2]，MSIN的值由运营商自行分配。

IMSI的格式由国际电信联盟（ITU）的E.212标准定义。

计费中心（Billing Center，BC），负责处理由VLR和HLR生成的收费单（toll tickets），并为每个签约用户生成一个账单。它也负责为漫游签约用户生成账单数据。

短信中心（Short Message Service Center，SMSC），支持短信的发送和接收。

合法监听功能，根据美国法律，也被照搬到许多其他国家，尤其是欧洲，所有的电信设备都必须提供设施用于对被选中用户的呼叫进行监听。必须有一定程度的支撑，以便它能被建立在任何不同的网元中。按照相关的美国法律，“合法监听”的概念也被称为“通信协助执法法”（Communications Assistance for Law Enforcement Act，CALEA）。合法监听的实现基本和电话会议的实现类似，比如当A和B在相互谈话时，C可以进入这个通话中，并静默地收听。

7.1.2　GSM加密算法的安全性

GSM的设计具有中等安全水平。系统设计使用共享密钥用户认证，用户与基站之间的通信可以被加密。演进的UMTS引入可选的USIM，使用更长的鉴别密钥保证更好的安全，以及网络和用户的双向验证。GSM只有网络对用户的验证（单向验证，而不是双向验证）。虽然安全模块提供了保密和鉴权功能，但是鉴权能力有限，而且可以被伪造。

每个移动站，即移动电话，都配有一个用户识别（SIM）卡，在SIM卡中存有用户在GSM系统中的注册信息，包括：给用户特定且唯一的身份标志的国际移动用户识别码（IMSI）、用户电话号码、鉴权算法（A3）、加密密钥生成算法（A8）、个人识别号（PIN）、单个用户鉴权密钥（K_i）等。在GSM移动电话中又包含加密算法（A5）。

为了安全GSM使用了多种加密算法，其中主要有三种加密算法：A3是鉴权算法，A8是密钥的约定算法，A5是用于加密的流密码。GSM规范并没有指定A3和A8的具体专用算法，仅仅指定了A3和A8算法的外部接口。加密算法的原型可由使用人员独立选择。值得一提的是，这两个算法的原型很多人使用GSM备忘录（MoU）里的例子，称为COMP128，该名称虽然没有公开过，但是却出现在很多作者的文章中，比如Briceno、Goldberg、Wagner等。

1.GSM三种加密算法简介

A3算法提供了手机到网络的鉴权，A8是产生密钥Kc的算法，这些加密算法是基于用户专用的密钥Ki的。K_i 被固化在用户的SIM卡内，GSM网络端的K_i 放在与用户号码相对应的用户归属位置寄存器（HLR）内。GSM没有专门规定K_i 的长度，而是留给运营商去选择。但是，通常Ki是128位长的密钥。用户对网络的鉴权使用A3算法，算法流程：网络向用户提供128位的随机数RAND，用户计算出32位长的相应SRES＝A3(K_i ，RAND)，并发送SRES到网络，网络检测它是否有效。

密钥K_c用A8算法获得：$K_c=A8(K_i, RAND)$，与上文中的A3计算过程相比，此时就可以发现，A3算法和A8算法的参数是一样的。实际上，在大多数情况下，它们是一个算法的两个输出结果，即SRES和K_c，因此一般总是用A3/A8算法来说明。A3/A8算法的实际使用流程如图7-1所示。

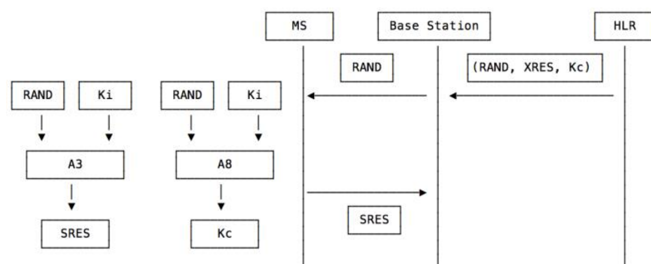


图7-1　A3/A8算法的实际使用流程图

GSM使用A5算法来保护数据安全，其中密码算法分别是A5/1、A5/2和A5/3算法。A5/1和A5/2算法是基于移位寄存器的序列密码算法，其中A5/1算法是在欧洲范围使用的强力算法；A5/2算法是A5/1算法的弱化版本，主要用于出口欧洲和北美之外的国家。A5/3算法是基于分组密码算法的，目前尚未启用。

A5/1算法（以下简称A5/1）是欧洲数字蜂窝移动电话系统GSM中采用的流密码加密算法，用于对电话手机到基站线路上的话音和数据加密。它的输入为86bit密钥，分为64bit会话密钥K_c和22bit帧序列号F_n，生成的114bit密钥流和114bit明文“异或”产生114bit密文，或者和114bit密文“异或”产生114bit明文。它工作在流密码的计数器模式下。

A5/1内部由三个线性反馈移位寄存器（LFSR）R0、R1和R2组成，级数分别为19、22、23，抽头数分别为4、2、4个，将R0、R1和R2的“异或”作为位输出。其设计思想是采用“三个LFSR互相控制时钟”的结构。A5/1有三个钟控输入，分别为每个LFSR的中间位，和三个钟控输出，分别控制每个LFSR的停/走。钟控机制采用择多逻辑，在每一轮中时钟驱动两个LFSR移位。若两个或三个控制比特值为“1”，则控制比特为“1”的寄存器移位；若两个或三个控制比特值为“0”，则控制比特为“0”的寄存器移位。这种不规则的互钟控结构使得A5/1的性能非常优良，可通过所有已知统计检测。

A5/1的加解密过程是相似的，每一帧密钥流按照如下方式产生。

（1）初始化：首先将三个移位寄存器全部清0；然后在64个时钟周期内，将每个移位寄存器都移位64次（不带钟控），并在每次移位后将密钥K_i（i＝0～63）置入每个移位寄存器的最低位；最后在22个时钟周期内，将每个移位寄存器都移位22次（不带钟控），并在每次移位后将帧序列号Framei（i＝0～21）置入每个移位寄存器的最低位。

（2）产生输出密钥流：将三个移位寄存器钟控移位100拍，丢弃输出；然后互钟控输出228bit，作为双向通信中使用的密钥流。

A5/1算法结构示意图如图7-2所示。

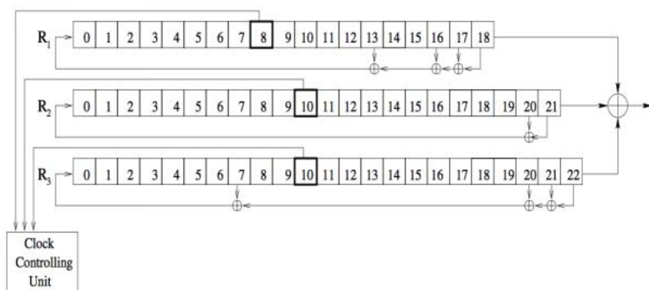


图7-2　A5/1算法结构示意图

A5算法输出228bit的密钥流，其中前114bit用于加密网络到用户的下行链路，后114bit用于加密用户到网络的上行链路。加密是简单地将信息与密钥流“异或”。每一帧通信用一个新的密码段K_c加密。

在A5/1的初始化过程中对密钥和帧序列号采用了线性填充，本意是使三个移位寄存器的内部状态与每一个密钥和帧序列号比特都有关，但是却降低了安全性，可能会遭受相关攻击而产生畸变的A5/1密钥流，从而暴露其统计特性。

此外，A5/1和A5/2这两种加密算法的严重漏洞都已经被发现，例如一个单一密文攻击可能实时地中断掉A5/2，不过系统设计时支持多种不同的算法，这样运营商可以更换一种安全等级更高的算法。

A5加密算法是GSM规范的一部分，由于保密，算法本身的技术细节并未公开。虽然最初该算法秘而不宣，但是1994年该算法被泄露，并且1999年该算法被Marc Briceno用逆向工程破解。2003年，Barkan发表了几篇关于攻击GSM加密算法的文章，A5/2算法可以轻易被破解。2007年，Bochum和Kiel大学创建了一个基于FPGA的加密

加速器项目（COPACOBANA），该项目被用来破解A5/1和A5/2算法。2009年，黑客Karsten Nohl发起了一个通过彩虹表方式破解A5/1算法的项目。

下面详细介绍GSM A5/1加密算法的漏洞。

2.A5/1算法漏洞

A5/1算法一经出现就成为研究的热点。近20年来，比较有代表性的是基于解线性方程组的猜定攻击算法和基于时空折中攻击的方法。但是由于通信的实时性要求，上述方法还未产生实质性威胁。直至2009年，Karsten Nohl公布了利用彩虹表破解的方法。Karsten Nohl使用4块GPU加速卡在一个月时间内生成2TB的数据表，可以达到在两帧已知明文的攻击条件下，使用两块GPU加速卡在5s内完成破解。这就使得GSM的使用受到严重的威胁。但在相关的论文中并没有对基于彩虹表的时空折中模型进行详细的描述，并且对参数的选择没有给出说明。

在实际攻击环境中，能够得到的数据是唯密文数据。但是在通信过程中，尤其是在控制信道中，在移动设备和基站交互的过程中会出现已知传输内容的情况，因此可以得到已知明文的数据条件。并且在实际攻击过程中利用的数据越少，所形成的实时性越高，攻击的效果和造成的威胁就越大，但同时对攻击方的要求也就越高。

在对A5/1算法进行攻击时，只要能根据密钥流数据得到某一时刻的寄存器状态，那么就容易利用Golic，J在2000年的Cryptanalysis of three mutually clock-controlled stop/go shift registers文章中提到的方法，根据已知IV向量值递推得到初始会话密钥K_c。通俗地说，就是对于A5/1数据我们需要捕获被攻击方发送和接收的System Information Type5消息，并且对传送的数据进行“异或”操作，而后通过彩虹表计算出加密的K_c值，再通过该K_c值对接收的数据进行解码，还原成用户可读的数据。对A5/1加密破解感兴趣的读者，可以翻阅针对A5/1时空折中攻击模型的参数选择研究的相关论文，此处不再详述。

但是在我国，由于种种原因，实际上大部分地区并未启用GSM加密，也就是在通信过程中，并未采用A5/1算法对空中往返的数据进行加密，而是采用A5/0，即以明文形式进行传输。下面就针对我国GSM通信安全漏洞常见的攻击方式进行详述。

7.1.3　GSM攻击

目前，国内攻击GSM的方式主要有两种：主动攻击和被动攻击。

所谓主动攻击，即攻击者会伪造成基站（BTS），然后发送诱导信号，引诱被攻击者连接到非法基站。根据我们之前对GSM协议中A3鉴权算法的介绍可知，GSM属于单向鉴权，因此移动台只能被基站鉴权，而无法对基站进行鉴权，因此用户很容易被诱导至伪基站进行数据通信。当被攻击者进行数据交换，比如拨打电话或者发送短信时，攻击者通过伪造的基站对其发送的数据进行劫持、篡改或监听以达到攻击目的。

而所谓被动攻击，即攻击者不会主动向被攻击用户终端发送诱导信号，而是监听基站与移动台之间传播的广播信号，并且对信号进行解密以达到侦听的目的。

主动监听与被动监听的主要区别在于，主动监听不但可以监听到被监听人的来往数据，还可以动态篡改来往数据；而被动监听只可以监听数据而无法直接篡改。

1.GSM主动攻击

如图7-3所示，攻击者通过USRP或者专用小型基站建立起私有的基站系统，并且与攻击者的电脑连接起来。攻击者在电脑上利用开源软件架设虚假的BSC、SGSN和GGSN服务，并且连接到互联网上。

攻击者通过信号干扰器对真实的基站信号进行干扰，诱导用户连接到虚假基站进行注册。之所以可以对移动台进行诱导，是因为GSM协议中只定义了BTS对移动台的身份验证操作，并没有要求移动台对BTS进行身份验证操作。因此不管是真实的BTS还是伪造的BTS，对于移动台来说并没有差异，它们只寻找其周围信号最强的移动台进行注册。用户一旦注册到虚假基站后，其所有数据就会被监听。

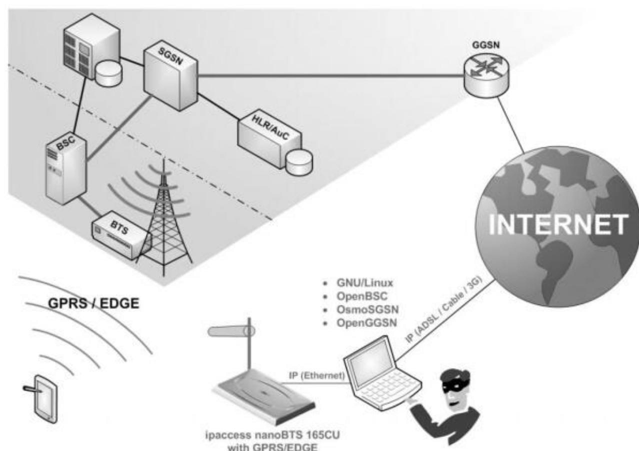


图7-3　GSM主动攻击结构图

当用户发起通话或者数据传输时，虚假基站会强制要求移动台采用A5/0加密方式（即不加密）进行数据传送。此后，移动台发送和接收的数据均以明文进行传递，攻击者则可以将用户传递的数据还原成明文查看或者收听。若此时用户正在使用GPRS访问网络资源，则攻击者可以监控用户的网络流量，在用户进行网银操作或登录账户等敏感操作时，记录用户的所有操作，造成其财产损失，甚至将用户的数据请求转发到已经存在的攻击服务器上，诱导用户下载木马等程序，以便进一步对用户进行控制。

此前猖獗一时的伪基站短信大部分采用的是USRP等开源设备进行伪基站架设，即硬件使用USRP，软件采用OpenBTS充当平台，诱导GSM手机接入伪基站的网络，并对辐射范围内的GSM协议手机进行诈骗，或者发送骚扰短信。

2.GSM被动攻击

被动攻击是指通过接收机（通常是USRP或者改装过的移动台）接收与用户相同的基站发送出来的信号，并且对用户发送的BURST数据进行分析和解码，最终还原成可读的明文数据。一般国内的语音和数据信号采用A5/1和A5/0加密方式。对于A5/0加密方式不需要进行额外的处理，语音和数据信号采用明文传输；而对于A5/1数据则需要按照前文描述的方式事先生成彩虹表进行枚举攻击。如图7-4所示是针对被动监听方式常见的攻击应用场景。



图7-4　针对被动监听方式常见的攻击应用场景

这里采用对常见的移动台（摩托罗拉C118）进行修改的方式，对移动台所在的位置进行被动监听，以达到语音通话侦听、短消息侦听、GPRS数据侦听的目的。

下面对GSM被动攻击原理进行介绍。

OsmocomBB（Open Source Mobile Communications-Baseband）是为手机基带处理器开发的一个免费开源的固件。OsmocomBB？从底层对GSM客户端的协议栈进行了重构，开发了开源的固件，专用于被动监听。把这个固件刷入C118手机，并与PC进行数据交互，就可以达到被动监听手机附近的无线通信的目的。

- 准备编译环境

这里采用的Linux环境为Ubuntu12.04x86（交叉环境不支持64位系统），其他系统方法一样，命令会有些差别，请自行分辨。

安装各种依赖库：

```
apt-get install build-essential g++ gcc make automake libtool libusb-dev pkg-config git
```

安装libosmocore：

```
git clone git://git.osmocom.org/libosmocore.git
```

```
cd libosmocore/
autoreconf -i
./configure
make
sudo make install
```

编译osmocom-bb.git，细节可以参考https：//srlabs.de/gsm-map-tutorial/网址。

下载代码到~/cell_logger：

```
mkdir -p ~/cell_logger
cd ~/cell_logger
git clone git://git.osmocom.org/osmocom-bb.git
```

下载ARM交叉编译环境：

```
wget http://gnuarm.com/bu-2.15_gcc-3.4.3-c-c++-java_nl-1.12.0_gi-6.1.tar.bz2
tar xf bu-2.15_gcc-3.4.3-c-c++-java_nl-1.12.0_gi-6.1.tar.bz2
```

准备OsmocomBB‘s burst_ind branch：

```
cd ~/cell_logger/osmocom-bb
git checkout --track origin/luca/gsmmap
```

编译OsmocomBB：

```
cd src
export PATH=$PATH:~/cell_logger/gnuarm-3.4.3/bin
make
```

确认手机处于关机状态，刷入固件，不同设备需要刷入不同的固件。这里的例子是针对摩拉罗拉C118的，把带有2.5mm耳机插头的线一头连接手机，另外一头连接USB2TTL模块，在PC端运行如下命令：

```
cd ~/GSM/osmocom-bb/src/host/osmocon/
sudo./osmocon -m c123xor -p \
/dev/ttyUSB0 ../../target/firmware/board/compal_e88/layer1.compalram.bin
```

按下手机开机键，确认进入osmocom-bb系统，如果刷入正确，

则可以看到终端有提示刷入成功的输出代码。另外，手机上也会显示如图7-5所示的提示。



图75　C118刷入osmocom-bb成功提示

如果刷入错误，则多半是由于C118机器年代久远、耳机接头接触不良导致的。为了提高刷入成功率，一般都直接拆掉手机外壳。

刷入成功后，再打开一个终端，执行如下指令进行基站扫描及同步：

```
sudo ~/cell_logger/osmocom-bb/src/host/layer23/src/misc/cell_log -O
```

看到如下输出则说明扫描到可用的基站：

```
ARFCN 117: tuning
ARFCN 117: got sync
Cell ID: 460_1_03EE_B130
cell_log.c:248 Cell: ARFCN=117 PWR=-62dB MCC=460 MNC=01 (China, China Unico
```

这时，说明该手机已经和中国联通当地基站同步了，频道号是117。

执行如下指令抓包：

```
~/cell_logger/osmocom-bb/src/host/layer23/src/misc/ccch_scan -i 127.0.0.1 -a 117
```

这条指令参数中的117，即来源于基站扫描结果中的117。

此时，手机监听到的并且传送回来的所有GSM数据即通过本地127.0.0.1地址的4729端口被抓取，我们可以用wireshark解析所抓取的数据包，运行如下指令即可：

```
wireshark -k -i lo -f 'port 4729'
```

在wireshark的过滤器中简单地输入gsm_sms即可过滤出所监听到的短信，相对应的，在wireshark中也可以直接过滤出语音和GPRS数据。如图7-6所示即为捕获到的短信数据包：

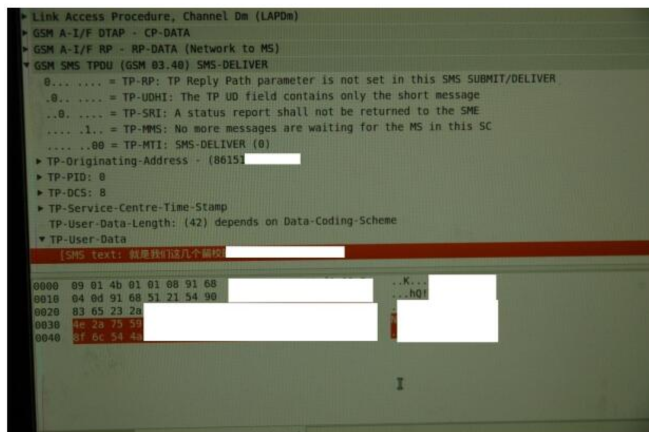


图7-6　短信数据包捕获示例

此外，由于空中信道中最多有7个业务时隙，一部手机只能对应一个时隙，所以有人对ccch_scan的源码进行了Patch，可以一台PC支持多部手机进行数据捕获，需要使用Sylvain Munaut的DSP Patch版本，git地址：http://cgit.osmocom.org/osmocom-bb/？h＝sylvain%2Fburst_ind，感兴趣的读者可以在该地址进行源码克隆并编译。如图7-7所示即为在网上流传较广的多机抓包实例图。

早期对于GSM算法我国大部分地区采用的是不加密措施，即C118监听到的数据即为明文。但截至本文完稿近期（2015年12

月），部分地区逐步对GSM进行了算法替换，即采用A5/1算法加密，这些算法一直内置于电话内部，只不过一直未启用。因此在采用C118进行监听尝试时可能会遭遇到数据加密的问题。此时可根据前面描述的A5/1算法采用彩虹表的方式进行破解，只是这种方式相比直接监听显然更加耗时、耗力，本文此处不再赘述。

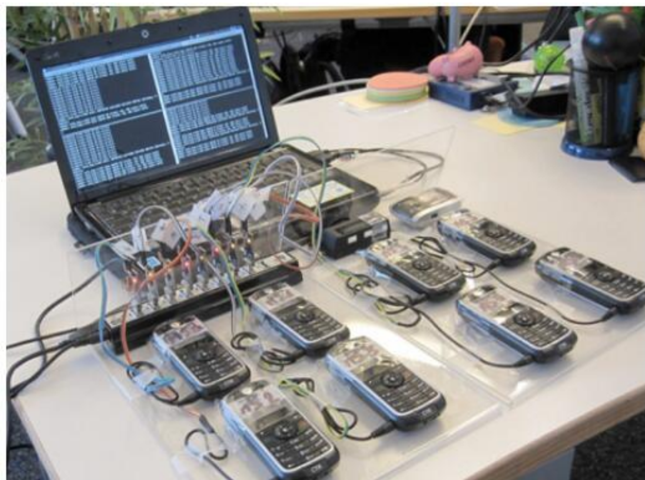


图7-7　多机抓包实例图

7.2 IMSI Catcher

7.2.1 什么是IMSI Catcher

IMSI Catcher是一个伪基站，用于跟踪移动电话用户的移动。其原理是抓取附近手机的IMSI（International Mobile Subscriber Identity），因此可以做到跟踪某个目标手机，或者记录哪些手机在这一地点出现过。IMSI Catcher原理示意图如图7-8所示。

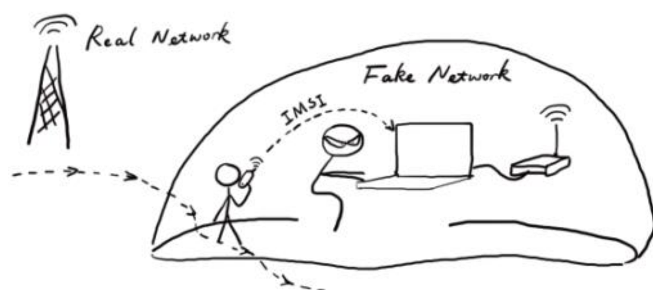


图7-8 IMSI Catcher原理示意图

IMSI Catcher被用在一些国家的执法和情报机构里，但这是建立在公民隐私和国家安全相平衡的基础上的。而非国家机关的其余人使用IMSI Catcher，则是非法行为，它不仅侵犯了个人隐私，还干扰了运营商网络。

尽管这种设备实际上只是一个由有恶意的管理员操作的修改后的基站塔台，很难被维持专利权，但是还是在2003年时被Rohdeamp;Schwarz公司申请专利保护并首次进行商业化运作。

GSM规范中要求，手持端需要被网络端进行身份校验方可接入网络，但是并没有要求手持端对网络端进行合法验证。普通的IMSI Catcher即是利用这个众所周知的安全漏洞进行工作的。这种IMSI Catcher伪装成一个GSM基站，当其所覆盖范围内的GSM手持设备试图连接至IMSI Catcher时，它会记录下所有这些设备的IMSI号码。记录完之后，就把手持设备踢出自己的网络，让它回到运营商的网络中

去。

此外，当手持设备连接至IMSI Catcher时，它可以强制该设备使用未加密模式（A5/0模式），或者使用易于破解的加密方式（A5/1或A5/2模式），从而轻易监听通话数据并将其转换为语音。

在美国，便携式的IMSI Catcher经常被用于攻击持有人附近的手持设备，是执法机构使用的一个重要设备；此外，IMSI Catcher还经常被用在没有获取到搜查令的执法行动中，用于追踪相关人员。当然，IMSI Catcher不仅用于执法，还经常用于搜寻救援失踪人口。

除了GSM网络以外，3G和4G网络也存在类似的问题，都能够实现IMSI Catcher。下面我们针对这些场景下的IMSI Catcher来分别进行详细说明。

7.2.2　GSM环境下的IMSI Catcher

1996年，德国公司Rohdeamp;Schwarz在慕尼黑推出了第一款IMSI Catcher GA090，IMSI Catcher最初的想法是用于通过强迫手机传输IMSI从而标识出手机所在地理位置，通过与网络运营商合作可以做到利用手机号定位到某个手机用户的精确位置。1997年，更为成功的GA900型号产品不仅可以用于识别用户，还可以监听用户拨出的电话。

在GSM中，IMSI Catcher主要依赖的漏洞就是单向鉴权，就像我们前面所讲的，移动端设备并不对基站端的可信性进行验证。IMSI Catcher利用这个漏洞，面向移动端设备伪装成一个基站。从理论上说，在信号强度达到25W的情况下，IMSI Catcher可以影响几公里区域范围内的手机。另外，GA900型号产品可以插入SIM卡，因此还可以将自身伪装成一个移动端，从而形成中间人攻击（MITM）。图7-9显示了这种场景下的GSM协议修改过程。

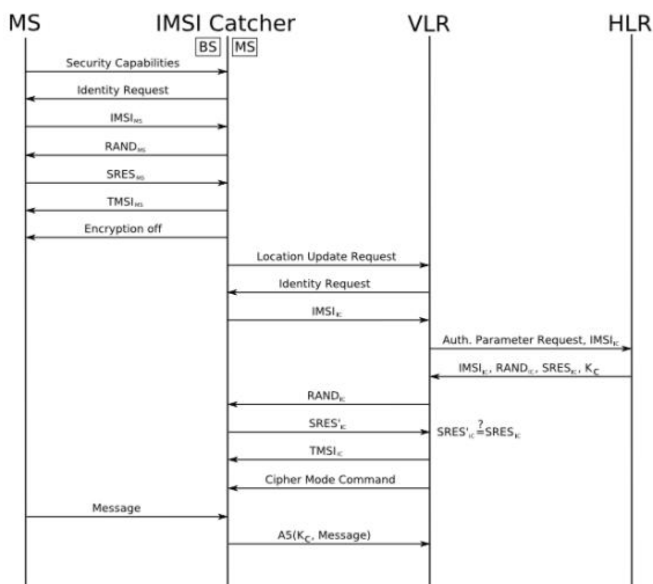


图7-9　GSM网络中的IMSI Catcher

1.识别IMSI号码

每一部移动手机都需要针对当前环境优化自己的接收信号。如果当前环境下有多于一个的可用的基站，那么手机会一直选取信号最优的那个基站。IMSI Catcher会伪装成基站，吸引所有影响半径内的指定运营商的手机去连接它，通过发送特殊的身份鉴权请求，IMSI Catcher可以强迫手机传输IMSI而非预定情况下的TMSI。

2.监听移动电话

在这个案例中，IMSI Catcher伪装成基站，同时又作为一个移动端与真实基站进行通信。在通过验证之后，IMSI Catcher利用基站拥有强制不使用加密算法的权限这个漏洞，要求手机与其连接时不加密通话流量；然后，它把真实手机发送来的明文信息加密后传送给真实基站。

由于IMSI Catcher采用一张SIM卡来建立常规连接，因此在同一时间内不能监听多于一个的拨出电话。那么为什么拨入的电话无法通过这个链路呢？因为运营商并没有办法直接寻找到掉入陷阱的手机。

此外，不依赖SIM卡的中间人攻击也是可行的。在验证过程中，IMSI Catcher简单地将验证数据在手机端和真实基站端相互传递，但是在这种情况下，因为IMSI Catcher并没有办法拿到加密密钥K_c，所以两边传输过程中的加密都必须被禁用。另外需要注意的是，向手机发送A5/0加密命令比较简单，但是由于GSM本身安全性的规定，试图强迫基站在连接时采用无加密模式是比较困难的。因此，使用SIM卡创建常规数据连接是一种比较简单的方式。

3．定位

由上面介绍可知，IMSI Catcher无法精确定位一部手机的位置，它只能验证指定手机是不是在预定范围内出现。所以，IMSI Catcher通常会配合一个角度非常窄的定向天线，用人工跟踪的方式，搜索手机信号最强的方向，从而定位手机的精确位置。

7.2.3　UMTS环境下的IMSI Catcher

在UMTS环境下，标准制定协会采用双向的合法性验证手段，即手机和基站会相互验证对方的真实性，因此，在GSM环境下可行的中间人攻击在UMTS场景下已经不再适用了。IMSI Catcher无法获得密钥K，因此也无法得到身份验证向量AV。因为这个原因，在UMTS环境下通过之前的方式是无法再捕获到IMSI的。

但是，在2005年，Ulrike Meyer和Susanne Wetzel描述了一种攻击方式，通过UMTS内嵌的GSM相关流程来进行攻击。同样是中间人攻击，但是分为以下3个步骤。

（1）需要预先获取IMSI或者一个有效的TMSI。这比较简单，因为在握手认证的初期阶段被移动端发送。

（2）模仿被攻击者连接至运营商网络。攻击人员首先将IMSI/TMSI发送至运营商网络，然后等待运营商基站将随机数RAND以及验证令牌AUTN发回，收到这些关键数据之后，断开与基站的连接（如图7-10所示）。RAND和AUTN的组合被留存以便稍后使用。

（3）IMSI Catcher伪装成GSM基站，然后将RAND和AUTN发送至被攻击者。由于这个时候与第二阶段时间间隔很短，所以这些用于验证的数据时效性很高，因此可以被移动端接收。攻击者无法验证返回的验证数据RES，但是这并不重要，接下来向移动端发送指令要求其采用GSM的A5/0加密模式进行通信（详见图7-11）。

由于无法同时面向基站模拟被攻击者客户端，所以为了转发实际的通信数据，伪基站需要建立一个固定的上下行连接。因此，攻击者本身需要拥有一个USIM连接到运营商方可进行攻击。

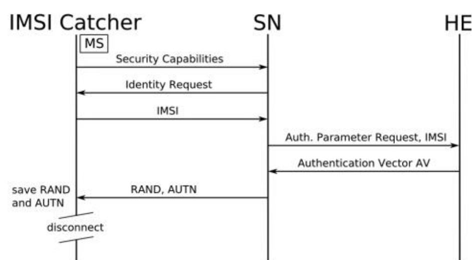


图7-10　攻击者获得当前可用的验证令

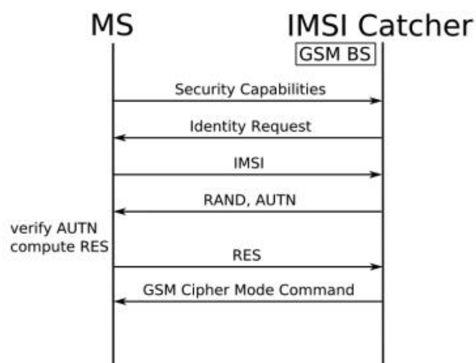


图7-11　攻击者伪装成GSM基站进行验证

7.2.4 LTE环境下的IMSI Catcher

LTE网络与UMTS网络类似，也是双向鉴权。但是IMSI仍然是可以获得的，因为无论2G、3G还是4G网络，在终端与网络建立连接的过程中，终端都不得不向网络发送自己的身份信息。

在2015年BlackHat Europe会议上，德国和芬兰的一个研究小组介绍了他们对LTE IMSI Catcher的研究—LTEamp;IMSI Catcher Myths。图7-12显示的是这个研究小组利用USRP和OpenLTE开源代码实现的低成本IMSI Catcher。



图7-12　利用USRP和开源软件实现的低成本IMSI　Catcher

LTE在连接初始化过程中，有一部分信令是不加密的，因此可以被利用。信令流程大致如图7-13所示。

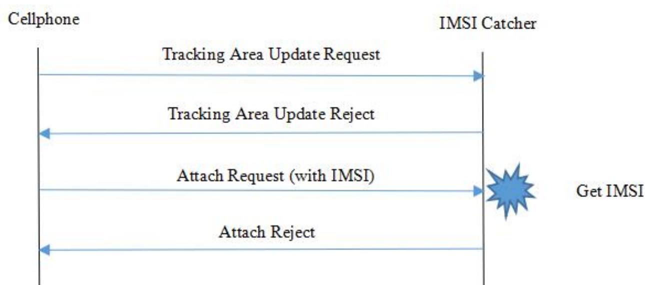


图7-13　信令大致流程图

当手机检测到一个更强的LTE基站时，就尝试连接到这个基站。手机会发起位置更新请求，此时携带的是TMSI。IMSI Catcher当然是不认识这个TMSI的，也不需要TMSI，它返回一个拒绝消息，要求手机发起Attach流程。于是手机只好发出Attach Request，这个请求会携带IMSI号码。这样IMSI Catcher就拿到了它想要的IMSI了，最后发一个Attach Reject消息把手机踢出去。

这里要特别强调的是，无论是Tracking Area Update Reject消息还是Attach Reject消息，都可以携带一个EMM Cause值，这个参数用来说明拒绝的理由。有哪些理由呢？下面举几个例子。

- IMSI unknown in HSS：我的HSS中没有你这个IMSI。
- Implicitly detached：你的连接已经被释放了。
- No suitable cells in tracking area：没有合适的基站给你用了。
- Network failure：我的网络坏了，没法为你服务。

这样的理由有很多。在3GPP协议24.301中有一节是Cause values for EPS mobility management，其中列举了所有的EMM Cause值。不同的EMM Cause值会触发手机产生不同的行为，使它发起Attach请求，或者使它停止连接到网络。这一部分信令，是LTE网络攻击中最常被利用的一点。

7.2.5 IMSI Catcher的缺陷

在使用IMSI Catcher时其本身有很多缺陷：

- 必须要确保指定跟踪对象的手机正处于待机模式，并且已经暂时没有与运营商基站建立连接。因为对于一部手机来说，没有必要再去联入伪基站。

- IMSI Catcher的信号越强，捕获到的IMSI号码越多，但关键是如何找到自己需要定位的那个IMSI号码。

- IMSI Catcher捕获到的所有手机基本都无法正常地与运营商核心网络建立连接。之所以说“基本”，是因为除了指定跟踪监听人员的电话之外，其余接入到IMSI Catcher的手机将无法通过IMSI Catcher建立起与运营商的正常通信链路。

- 此外，还有一些容易暴露的因素在里面。在大多数情况下，这种攻击行为无法很快地被被攻击者发现，但是，当连接采用非加密模式时，有些手机会在状态栏显示感叹号进行警示。另外就是在拨打电话时，由于网络权限由IMSI Catcher的SIM/USIM卡接管，所以被呼叫者无法看到主叫方的号码。当然，这同样也导致了被窃听的电话实际上是不会在电话账单上出现的。

- 在真实基站附近安置的IMSI Catcher大多是无法起作用的，因为一般情况下IMSI Catcher都无法做到信号强度比真实基站大。

7.2.6 Stingray手机追踪器

Stingray（黄貂鱼）是一款由Harris公司制造的比较有舆论争议的IMSI Catcher手机监控设备（见图7-14）。其最初是为军队和情报机构研制的。Stingray和具有类似功能的Harris公司的其他设备在美国各个州的执法部门得到广泛使用，后来在加拿大温哥华被使用。Stingray已经成为了执法部门对此类监听跟踪设备的通用名称。



图7-14　Stingray 实物图

Stingray是一款同时具有被动（监听数据分析）和主动（基站模拟）功能的IMSI Catcher。当运行主动模式时，Stingray会模拟无线载波小区基站设备，然后迫使周围的手机或者其他蜂窝设备连接它。Stingray系列的设备可以被安装在面包车内以及飞机或者无人飞行器上。在行业中便携式Stingray又被称为KingFish（见图7-15）。



图7-15　便携式Stingray（又称KingFish）实物图

1.主动模式操作

- 提取存储的数据，比如IMSI或者ESN。
- 将蜂窝协议元数据写入内存。
- 迫使提升信号发送功率。
- 增大无线传输信号的丰度。
- 监听通信内容。
- 追踪定位蜂窝设备用户。
- 进行DDoS攻击。
- 导出通信中的加密密钥。
- 为了实施DDoS攻击或者在主动模式下实施降级攻击而进行无线电干扰。

2.被动模式操作

通过对空中的无线信号进行分析，标记出合法基站，对基站覆盖范围进行精确测绘，从而进行基站分布调查。

下面我们对Stingray的主要和通用的功能进行解释。

在主动模式下，Stingray将迫使特定区域内各个兼容的蜂窝设备与其对应的运营商基站（如Verizon、ATamp;T等）断开连接，并且与Stingray建立新的连接。在大多数情况下，Stingray发送比周围真实基站的信号强度更大的信号进行压制即可。在蜂窝通信设备中一个通用的特性就是设备将会连接周围提供最强信号的那个基站。Stingray正是利用这个特性，将有限范围内的移动设备都吸引连接至其本身。

当Stingray将指定区域内所有适配的移动终端都吸引到自己身上时，Stingray的操作人员需要找到哪一个设备才是自己要监听的，这需要获取所有已连接设备的IMSI、ESN或者标识性数据。在这个前提下，IMSI或者其他类似的标识号码不是从第三方或者网络服务商那里获得的，而是Stingray直接利用相关协议从连接的终端下载得到的。在某些情况下，Stingray的操作人员会提前得知受监控人员设备的IMSI或者ESN信息，这时每一个设备连接至Stingray之后，操作人员都会获取设备的标识号码。一旦接收到的IMSI符合预定目标，Stingray即停止捕获其余设备，操作人员开始只对预定的监视目标采取措施进行监听或跟踪。

而在有些情况下，监控人员事先无法得知受监控人员的IMSI或者ESN信息，接下来要在预定区域内对这种目标进行监控，并且这种区域不止一个蜂窝设备。比如，监控人员隐藏在一堆抗议人群中，Stingray就被用于获取该区域内的所有IMSI或者其他身份标识信息，当手机信息被区分之后，这些信息就可以用于跟踪及监听了。此外，网络运营商也可以被合法要求向执法机关移交这些IMSI相关的账号信息，以便进行定位。

在被动模式下，Stingray配合一部测试手机，通过搜集一个基站的识别号码、信号强度、信号覆盖范围等关键数据，可以对基站进行分布调查。当进行基站信息调查之后，Stingray会模拟一部手机，接收并且分析基站发送的信号。

所获得的基站基本数据，如果与从无线网络获得的历史小区站点位置信息（historical cell site location information）一起使用，则可以被用来进一步锁定一个蜂窝设备过去的位置。历史小区站点位置信息包括与一个蜂窝设备连接过的所有基站的信息，以及每次接入的具

体时间。执法部门往往会从无线运营商那里获得历史小区站点位置信息，以确定一部特定的手机过去所去的地方。一旦获得该信息，执法部门就可以通过之前绘制好的基站分布地图来确定特定的蜂窝设备过去的路线图。

但是，确定的信号塔的信号覆盖范围会根据当天测定的时间、天气、物理障碍物的改变而改变。执法部门手中使用的基站分布地图也是用于通用情况下，而非精确绘制。由于这些因素，可以使用Stingray和一部测试手机来对历史小区站点位置信息记录中出现的所有基站进行更为精确的绘制。这种措施一般都是在历史小区站点位置信息测绘的当天或者同样天气的日子进行的。采用这种方式进行的测绘，可以更清楚地对相同条件下监控的蜂窝设备所在的信号覆盖区域进行绘图。

7.2.7 IMSI Catcher Detector

IMSI Catcher并非是不可对抗的，在GitHub上有一个开源的Android项目可以用于监测附近的IMSI Catcher，地址为http://secupwn.github.io/Android-IMSI-Catcher-Detector/。

该项目又称AIMSICD，旨在监测手机连接的基站是否为“伪基站”（IMSI Catcher），以及监测手机当前是否处于伪基站的MITM攻击下。

对于当前手机所处的网络环境，AIMSICD进行评估后进行分级，按照危险程度分为5级，并在地图上标出警示图标。警示图标如图7-16所示



图7-16　警示图标

如图7-17所示是AIMSICD正常工作时的截图，可以看到其读取设备信息、SIM卡信息、已连接基站信息，对网络威胁状况进行评估，并标记在地图上，进行可视化概括。

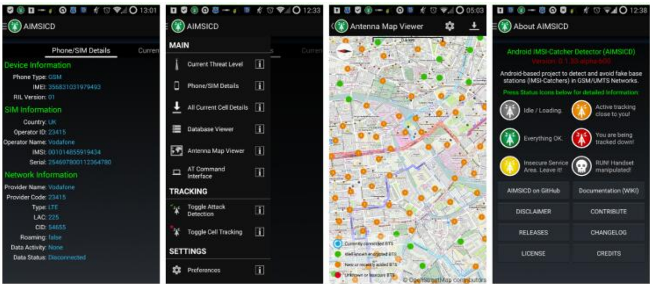


图7-17　AIMSICD正常工作时的截图

在官方描述中，AIMSICD主要通过监测如下信息进行伪基站判别：

- DBe连续性检查。
- 阻止Silent App安装。
- LAC/Cell ID连续性检查。
- 周边基站信息检查。
- 信号强度监测。
- 监测是否有Silent短信。
- 监测Femtocell（飞蜂窝）是否存在。

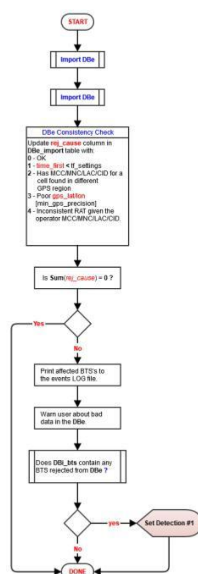


图7-18　数据检查流程图

当从OICD和MLS等导入外部BTS DB时，DBe连续性检查可以过滤出不正常数据以及违法BTS的相关信息。具体工作时，先从OCID导入所有的DBe数据，以一定规则来检查数据，检查的严格程度可以在设置中人工指定，根据数据的恶意程度，程序将对其进行分级标识。图7-18显示的是数据检查流程图。

在对DBe数据进行评级之后，AIMSICD会对手机连接到的基站的LAC（Location Area Code，区域代码）进行检查。如图7-19所示，通过LAC将一组基站塔（具有不同的ID）连接在一起。

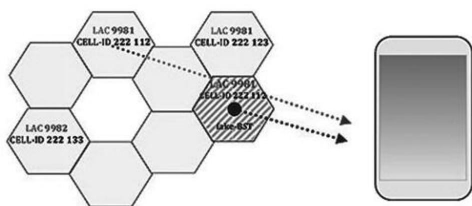


图7-19　IMSI Catcher模拟BTS效果图

图7-19中的IMSI Catcher模拟一个真正的BTS，然后发送了比原来合法的BTS更大功率的信号，因此将周围的手机吸引到伪基站上。但是在连接建立之后，双方依然使用TIMSI（Temporary IMSI/临时IMSI）。

图7-20显示了获取IMSI的一种可能的方式，即更改BTS的LAC号码，导致位置更新过程重新初始化。如果MSC（BTS的集群控制中心）同样根据位置更新而改变了，那么手机就必须发送IMSI；即使位置更新失败了，也同样要发送IMSI。

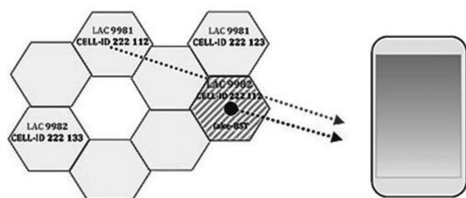


图7-20　IMSI Catcher更改LAC号码模拟BTS效果图

在AIMSICD中，我们拥有LAC和Cell ID的值，如果Cell ID和LAC随时间的改变而改变，那么AIMSICD将会显示黄色警告；如果其改变不止一次，那么将显示红色警告。这只会在IMSI Catcher进行IMSI捕获的过程中发生并告警，而不是在通过IMSI Catcher拨打电话时，所以不用担心为了监测伪基站而实际要去变成伪基站的受害者，方可进行有效监测。

在通常情况下，手机的TMSI号码存储在SIM卡中，一般不会改变，除非基站主动要求，或者在不同的网络环境下启动设备。一般情况下，由于BTS无法在VLR中找到最近的TMSI信息，所以会要求一个新的TMSI。但是，这种主动要求更新一般也会发生在IMSI Catcher强迫手机进行位置更新时。因此，当手机的TMSI号码突然发生改变时，则有可能手机已经被IMSI Catcher跟踪了。

此外，当手机并没有大范围移动或者加速运动，而收到的信号强度却急剧增大，并且当前连接的基站参数（LAC/Cell ID）保持不变时，这就预示着此手机可能掉入了IMSI Catcher的工作范围之内。

但是，信号强度不是一个太好的指示IMSI Catcher的参数，至少不是所有手机在正常情况下都有一个平缓的信号变化过程。把手机放在桌子上，然后在手机上方挥动一下手，都有可能比较大地更改接收信号强度。实际上，环境变化并不是信号强度这个参数不准的唯一原因，信号强度还和设备有很大的相关性。

因此，信号强度这个指标不会单独使用，而是与其他指标一起使用来进行IMSI Catcher的判定。

7.3 Femtocell安全

7.3.1 Femtocell简介

提到蜂窝网络安全，不得不说的就是Femtocell，即在国内我们称之为飞蜂窝或者家庭基站的硬件。

家庭基站（Femtocell，又译为毫微微蜂巢式基地台），最初叫接入点基站（Access Point Base Station），是一个小型的蜂窝基站，一般被设计为在家庭室内或小的商业机构中使用。通过宽带接入（如DSL、有线电缆、光纤）连接到运营商的网络，可以整合2G、3G、WiFi于一机。

2008年全球移动通信大会（Mobile World Congress，MWC）首次展出了Femtocell技术，得到运营商的青睐。“Femto”本意是指10的-15次方，又称为毫微微。家庭基站最早在欧洲流行，是一种超小型手机基站设备，相比Microcell基地台（通常覆盖范围在2公里以内）以及Picocell微型基地台（通常覆盖范围在200米以内）更小型，家庭基站的覆盖范围约为12米。在部分情况下，Femtocell可以提供更大范围的覆盖，以满足办公环境的需要。

家庭基站可以将3G用户家中的ADSL或光纤上网带宽转换为无线3G信号，适用于家庭及办公室环境。

家庭基站结构简单，仅基于IP协议，发射功率为10~100mW，也可以在提供3G网络服务的同时提供WiFi功能，相对于传统基站成本低廉，使Femtocell成为极具吸引力的解决方案。

当家庭用户所在位置的基站信号不强，无法正常、流畅地使用手机拨打或接听电话，并且运营商由于种种因素短时间内无法解决基站发射信号的问题时，一般会向用户发放家庭基站这种硬件设备，用于解决家庭用户手机信号的问题。具体使用时，一般都是将家庭基站的网络接口连接至家庭用户的上网环境中，如路由器或者光猫等。家庭基站的大小一般与家用路由器相仿，当家庭基站启动之后，将相当于一个小基站发送/接收手机对应的制式信号，并且将解析后的数据通

过家庭网络环境传输至运营商核心网，使家庭用户享受一定范围之内的正常运营商网络，从而正常进行收发短信、拨打/接听电话、手机上网等。

7.3.2 家庭基站的攻击面

家庭基站网络结构图如图7-21所示。

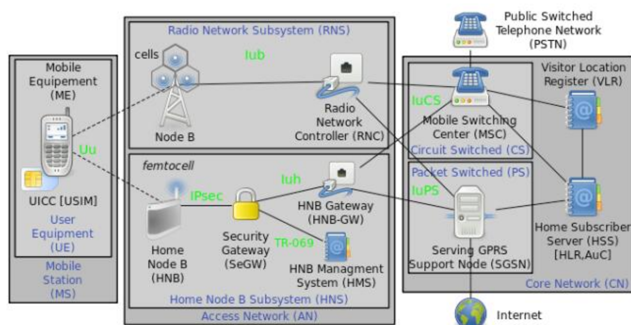


图7-21　家庭基站网络结构图

由以上描述可知：

- 家庭基站是一个合法的小基站。
- 家庭基站在用户手中掌控。
- 家庭基站可以向其影响范围之内的手机发送短信或拨打电话。
- 家庭基站可以接入运营商核心网。

因此，我们可以看到，一个有漏洞的家庭基站，是黑客手中的一款可用于监听、伪造电话、短信流量的硬件利器。下面我们就对近年黑客大会上出现的几个针对家庭基站进行攻击的案例进行分析。

7.3.3 CDMA Femtocell漏洞综合利用

NCC Group在BlackHat2013及DEFCON21上的Traffic Interceptionamp;Remote Mobile Phone Cloning with aCompromised CDMA Femtocell演讲[注]中，讲述了有关Verizon分发的三星SCS-2U01家庭基站（如图7-22所示）的安全漏洞，完整分析、逆向了此款家庭基站的固件，最后将其打造成一款可供监听、伪造、克隆的黑客工具。有关此次的演讲和PPT是一份关于家庭基站物理攻击及固件分析不可多得的有价值的参考资料。



图7-22　SCS-2U01实物图

在本案例中，NCC Group的研究人员获得了一款美国Verizon运营商的三星家庭基站，此款家庭基站的制式为CDMA。此款家庭基站激活后，与我们之前描述的家庭基站的工作方式一样，接入运营商网络后，正常接管其工作范围之内的基站信号进行数据交互。具体的接入方式如图7-23所示。

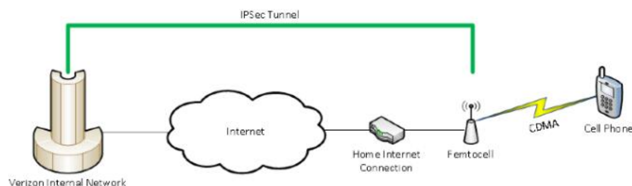


图7-23　三星家庭基站接入网络方式

作为黑客掌握的硬件设备，此款家庭基站的板载操作系统为Linux，所以第一步就是找到接口，root该款设备。在分析人员拆解并仔细研究该款设备的硬件配置后，他们发现设备底部的HDMI接口实际上与设备的串口调试引脚相连，如图7-24所示。



图7-24　设备底部的HDMI接口实物图

在接入HDMI线之后，研究人员切断并改造线缆，将其连接至USB FTDI（USB转串口适配器），设置串口对应的波特率和停止位。开机后，设备将会从串口输出调试信息。在研究过程中，研究人员还使用DEFCON20的Badge改装之后作为RS232-USB转接适配器，如图7-25所示。

经过逆向，研究人员发现，设备对应的U-Boot在开机时如果接收到’sys\r‘字符串，将会停止启动，接收调试命令。因此，打断U-Boot开机，在开机命令中加入bash执行，即可在设备启动后直接获得其root权限。由于打断了设备原有的启动流程，开机后用于使设备发挥原有功能的自启动脚本都将无法自动运行，研究人员在分析系统启动脚本之后，逐个找到自启动脚本并手动启动。

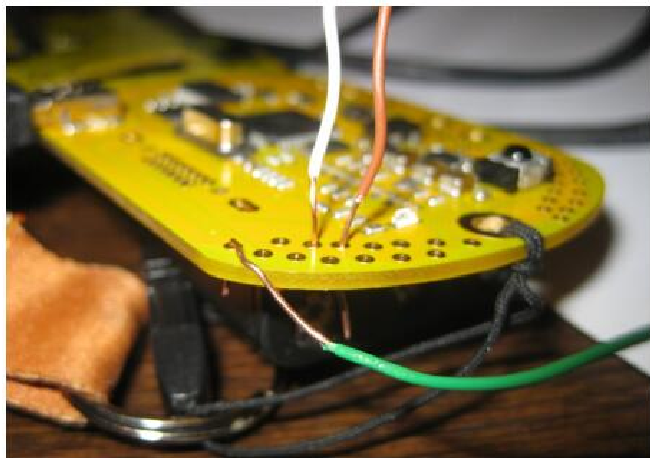


图7-25　DEFCON 20的Badge改装成的RS232-USB转接适配器

在系统正常工作后，使用Linux内置的命令，即可查找到系统主要进程、开放的网络服务、路由表、敏感的配置数据等，以便进行下一步工作。

这里我们就可以发现，在实际破解此类嵌入式设备过程中，获取root shell权限的重要性。获取到设备的shell之后，由内部进行网络流量、进程分析，甚至进行调试，远远比在外部进行黑盒状态测试要简单、方便、有效得多。而对于嵌入式设备，找到系统的调试接口（一般是串口），是对系统进行动态内部分析的重要一步。

在分析过程中，研究人员发现，Verizon公司为了保证网络通信链路上家庭基站回传的数据不可被监听，即采用了VPN进行流量隧道加密的措施。为了方便链路分析，研究人员进行资料搜索后发现，由于SCS-2U01使用了开源系统的套件，根据GPL协议，三星在其官网的相关链接下公布了包含有基于MontaVista Linux5的内核源代码。依据板载硬件，他们又在TI官网找到了包含有详细开发指导的芯片组开发套件。根据这些源代码信息，研究人员自己写出一套内核驱动插件，在设备启动之后加载驱动，更改系统路由表，将解密后的系统流量通过NC传出并进行进一步的流量分析。

解密后的流量大部分都是UDP报文，端口也不规律，在研究人员将其强制解析为RTP流量之后，发现语音数据编码为动态RTP类型，如图7-26所示。

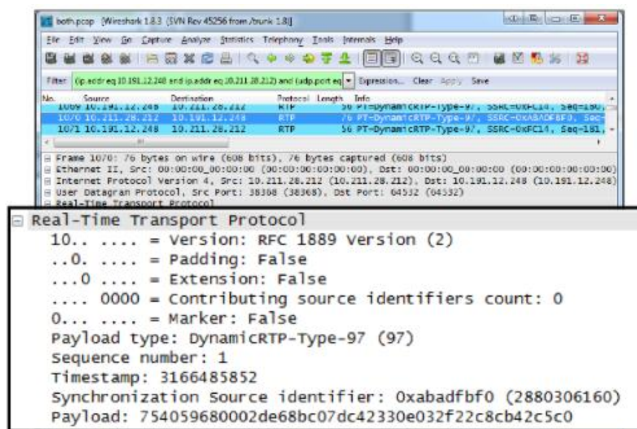


图7-26　UDP报文的Wireshark解析图（RTP协议）

SMS数据包采用了7位编码的方式，但是研究人员依然采用自己的脚本将其成功解析，如图7-27所示。

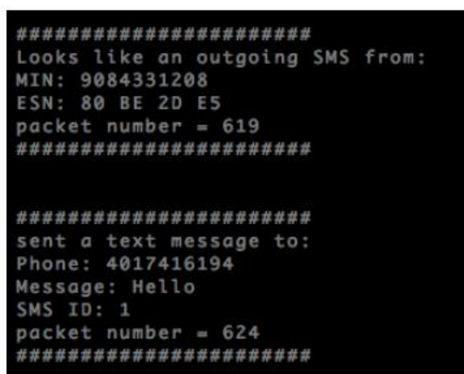


图7-27　SMS数据包监控脚本效果图

在经过解密后的明文报文中，研究人员分析出大量短信、电话及用户上网流量报文，图7-28和图7-29所示即为短信报文及网络流量报文示例。

由于掌握了流量分发方式，因此可以在流量中进行中间人注入，DNS劫持、HTTPS降级攻击、重定向URL等一切可以在普通PC环境下进行的中间人攻击方式经过一定修改之后，都可以在此攻击场景下使用。

No.	Time	Source	Destination	Protocol	Length	Info
734	16:41:59.797156000	10.211.157.206	10.184.98.60	UDP	42	Source port: 65246 Destination port: 6115
735	16:42:01.033494000	10.184.98.60	10.211.157.206	A9-SETUP	113	Source port: 6115 Destination port: 5603
736	16:42:01.230046000	10.211.157.206	10.184.98.60	UDP	578	Source port: 62078 Destination port: 6115
737	16:42:01.778752000	10.184.98.60	10.211.157.206	UDP	56	Source port: 6115 Destination port: 65246
738	16:42:01.825592000	10.211.157.206	10.184.98.60	UDP	56	Source port: 65246 Destination port: 6115
739	16:42:01.829454000	10.211.157.205	10.184.98.60	PPP LCP	73	Configuration Request
742	16:42:01.852154000	10.184.98.60	10.211.157.206	PPP LCP	119	Configuration Ack
743	16:42:01.884979000	10.211.157.205	10.184.98.60	PPP LCP	78	Configuration Ack
744	16:42:01.888488000	10.211.157.205	10.184.98.60	PPP CHAP	83	Challenge (NAME='SAMSUNG V.6 BSS', VALUE=0xb
749	16:42:03.571151000	10.184.98.60	10.211.157.206	PPP CHAP	112	Response (NAME='406[redacted]@vzw3g.com', VALUE
750	16:42:03.588989000	10.211.157.205	10.184.98.60	PPP CHAP	76	Success (MESSAGE='Welcome to SAMSUNG V.6 BSS
751	16:42:03.592504000	10.211.157.205	10.184.98.60	PPP LCP	78	Termination Request

```
▷ Frame 750: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
▷ Raw packet data
▷ Internet Protocol Version 4, Src: 10.211.157.205 (10.211.157.205), Dst: 10.184.98.60 (10.184.98.60)
▷ Generic Routing Encapsulation (CDMA2000 A10 Unstructured byte stream)
▷ PPP In HDLC-Like Framing
▷ Point-to-Point Protocol
▽ PPP Challenge Handshake Authentication Protocol
    Code: Success (3)
    Identifier: 1
    Length: 31
    Message: Welcome to SAMSUNG V.6 BSS.

0000  45 80 00 4c 00 00 40 00  3b 2f 29 6f 0a d3 9d cd   E..L..@. ;/)o....
0010  0a b8 62 3c 30 00 88 81  00 10 00 07 00 00 00 03   ..b<0... ........
0020  7e ff 7d 23 c2 23 7d 23  7d 21 7d 20 7d 3f 57 65   ~.}#.#}# }!} }?We
0030  6c 63 6f 6d 65 20 74 6f  20 53 41 4d 53 55 4e 47   lcome to  SAMSUNG
0040  20 56 2e 36 20 42 53 53  2e 9b 23 7e               V.6 BSS ..#~
```

图7-28　短信报文示例

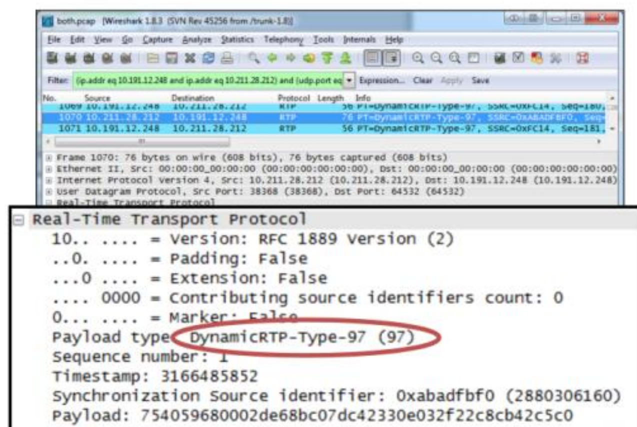


图7-29　网络流量报文示例

经过进一步的研究，研究人员发现，在此款设备的工作过程中，可以提取出合法手机连接至家庭基站后的ESN及MIN值，在对合法手机进行阻塞攻击之后，将获取到的合法ESN及MIN值克隆进违法手机内，连接至家庭基站，即可代替原先的合法手机进行正常的通话、接收短信，就此实现手机克隆。手机克隆过程如图7-30所示。

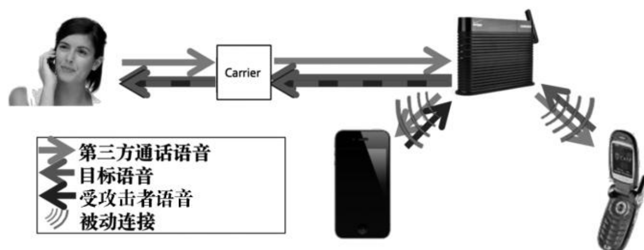


图7-30　手机克隆过程

在CDMA发展初期，每次用手机拨打电话时，都会直接传输未加密的MIN（Mobile Identification Number）、MEID（Mobile Equipment Identifier）或者ESN（Electronic Serial Number）数据到基站，用于对电话进行身份识别，这导致当时电话非法克隆非常猖獗。

后来，在CDMA技术中，用手机拨打电话时，会有一个叫作CAVE（Cellular Authentication and Voice Encryption，基站认证及语音加密）的流程，每部手机都有一个密钥A-Key，利用这个A-Key可以计算出两个衍生密钥，用于对电话拨打或接收的每个短信、电话进行合法性鉴定。此外，这两个密钥还用于对空中的语音信号进行加密。由于A-Key需要具有保密性，所以不会在空中链路中进行传输，但是衍生密钥会在空中链路中用于加密每个电话及短信。

在这款三星的Femtocell中，它的行为和正常的合法基站相同，唯一不同的是，在实际验证过程中，Femtocell不需要手机的MEID即可完成验证。在旧设备上一般使用的是ESN，新型设备一般使用的是(p)ESN，以代替MEID。这时我们就可以看到，Femtocell在验证过程中完全不用MEID，而只使用(p)ESN和MIN。而且最重要的是，在整个过程中，Femtocell根本就不需要CAVE！这也就意味着，一次标准的手机克隆只要知道(p)ESN和MIN就可以成功。

那么剩下的工作就是如何找到受攻击者的(p)ESN和MIN。

比较戏剧性的是，任何手机连接Femtocell时，都会直接将(p)ESN和MIN传送至Femtocell，只要手机落入Femtocell的捕获陷阱，而不需要任何物理接触。也就是说，不需要任何物理接触，Femtocell就可以克隆任何落入其影响范围之内并连接至Femtocell的手机。如图7-31所示为研究人员使用脚本过滤出数据流中的MIN和ESN值。

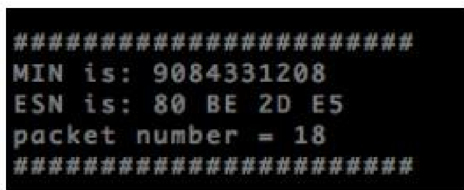


图7-31　MIN和ESN值

图7-32至图7-35展示了Femtocell是如何一步步克隆一部手机的。

（1）攻击目标手机落入Femtocell的移动范围之内。



图7-32　克隆手机第一步

（2）从受攻击手机上搜集MIN及ESN序列号，克隆进攻击者手机内。

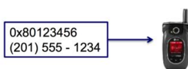


图7-33　克隆手机第二步

（3）攻击者手机与正常Femtocell连接上。



图7-34　克隆手机第三步

（4）手机上线，两部手机行为一致。



图7-35　克隆手机第四步

这种克隆产生的后果是：

- 假如受攻击手机被干扰或者关掉了，那么克隆用手机将会和受攻击手机一样，可以正常拨打/接听电话、收发短信。

- 假如受攻击手机和克隆用手机同时在线，那么拨出电话将会被强制丢弃，拨入电话是否有效要看哪部手机先振铃，短信只能接收不能发送；两部手机的机会是近似均等的（要看网络状况）。

此外，比较重要的一点就是，在常见的伪基站实现中，由于协议漏洞，所以基本都是GSM类型伪基站，市面上未有CDMA相关伪基站。然而，利用这样的合法设备，不法分子完全可以制作出一个低成本的CDMA伪基站进行诈骗、骚扰等非法行为，从而获利。

1. https：//www.nccgroup.trust/globalassets/newsroom/us/blog/documents/2013/femtocell.pdf返回

7.3.4 基于VxWorks的GSM Femtocell流量捕获器

奇虎360公司的无线硬件安全实验室（UnicornTeam）在2015年的DEFCON23会议上，介绍了针对国内移动公司免费发放的一款家庭基站进行的安全研究。与NCC Group的不同之处在于，这是一款基于VxWorks的设备。

VxWorks是美国风河（WindRiver）公司于1983年设计开发的一种嵌入式实时操作系统（RTOS），是嵌入式开发环境的关键组成部分。其具有良好的持续发展能力、高性能的内核以及友好的用户开发环境，在嵌入式实时操作系统领域占据一席之地。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通信、军事演习、弹道制导、飞机导航等。在美国的F-16、F/A-18战斗机、B-2隐形轰炸机和爱国者导弹上，甚至连一些火星探测器，如1997年7月登陆的火星探测器、2008年5月登陆的凤凰号和2012年8月登陆的好奇号也都使用到了VxWorks。

VxWorks在军事及国家安全领域的广泛应用也从侧面反映了其难以破解的程度，尽管在本案例中，开发人员的无安全意识行为造成了种种漏洞，但是VxWorks本身的安全性依然给安全研究人员带来不少困难。

在安全研究团队的Build AFree Cellular Traffic Capture Tool With AVxWorks Based Femoto演讲中，研究人员展示了如何通过社工免费得到这样一款设备，并对其进行物理攻击、破解固件及加密流量，最终将其修改为一款可以用来辅助进行蜂窝网络专属设备漏洞挖掘的“合法伪基站”。如图7-36所示为此款Femtocell实物图。



图7-36　基于VxWorks的Femtocell实物图

拿到此款设备后，研究人员首先对设备进行Web上的漏洞挖掘，获得了设备的固件及敏感配置文件。拆解设备后发现，设备的网络及硬件分为两个区域，其中一个区域是可用于正常上网的WLAN板，用于接通外部网络及分享WiFi信号；另一个区域为GSM板，用于发射GSM调制信号，并将解调处理后的信号传送给WLAN板回传至运营商核心网。通过对端口及IP地址的进一步扫描，研究人员发现设备开启了众多网络服务，如图7-37所示。

在对ftp和telnet服务进行用户密码爆破错误之后，研究人员注意到了名为wdbrpc的UDP端口，这个端口是VxWorks系统保留的用于远程调试的端口，在旧版本的VxWorks中，这个服务可以被攻击，H.D.Moore于2010年发表了关于VxWorks调试端口未授权利用的漏洞，利用这个漏洞可以进行内存提取等操作。然而，在此款设备中，这个漏洞“很不幸被修补了”。

```
root@am335x:/home/test# nmap -sT -sU 192.168.197.241

Starting Nmap 6.40 ( http://nmap.org ) at 2015-05-07 21:14 CST
Nmap scan report for 192.168.197.241
Host is up (1.0s latency).
Not shown: 997 open|filtered ports, 996 closed ports
PORT       STATE    SERVICE
21/tcp     open     ftp
23/tcp     open     telnet
80/tcp     open     http
514/tcp    filtered shell
50000/tcp  open     ibm-db2
69/udp     open     tftp
17185/udp  open     wdbrpc

Nmap done: 1 IP address (1 host up) scanned in 119.52 seconds
root@am335x:/home/test#
```

图7-37　设备GSM板开放服务

在软件层面上能被轻易利用的方式都没有了，研究人员转而从硬件上入手，在对硬件进行拆解及芯片组分析之后，很容易就发现了设备核心板上的调试端口，接入USB转串口设备之后，启动设备即得到如图7-38所示的信息。

```
                    VxWorks System Boot

Copyright 1984-2008  Wind River Systems, Inc.


CPU: TI OMAP-L138 - ARM926E (ARM)
Version: VxWorks 6.8
BSP version: 2.0/1
Creation date: Sep 26 2012, 10:26:36


Press any key to stop auto-boot...
 1
[VxWorks Boot]:
```

图7-38　设备启动后输出调试信息

在得到调试shell后，分析人员对设备的bootshell的启动参数进行详细分析，并且发现可以遍历设备装载的文件，如图7-39所示。

```
[VxWorks Boot]: ls /tffs0

Listing Directory /tffs0:
drwxrwxrwx  1 0       0            8192 Jan  1 00:01 ./
drwxrwxrwx  1 0       0            8192 Jan  1 00:01 ../
drwxrwxr-x  1 0       0            8192 Jan  1 00:00 common/
-rw-rw-rw-  1 0       0              12 Jan  1 00:00 startup.txt
drwxrwx-wx  1 0       0            8192 Jun 16  2015 user1/
drwxrwx-wx  1 0       0            8192 Jun 16  2015 user2/
drwxrwxr-x  1 0       0            8192 Jun 16  2015 wlanBackup/
-rw-rw-rw-  1 0       0          118781 Feb 13  2015 test.pcap
-rw-rw-rw-  1 0       0            1193 Feb 15  2015 ike.txt
-rw-rw-rw-  1 0       0            1195 Mar 16  2015 aaa.txt
-rw-r--r--  1 0       0             128 Mar 19  2015 imsi.cfg
[VxWorks Boot]: ls /tffs0/user1

Listing Directory /tffs0/user1:
drwxrwx-wx  1 0       0            8192 Jun 16  2015 ./
drwxrwxrwx  1 0       0            8192 Jan  1 00:01 ../
-rw-rw-rw-  1 0       0         4335633 Jun 16  2015 NodeB.zip
-rw-rw-rw-  1 0       0          638408 Jun 16  2015 appBooter
-rw-rw-rw-  1 0       0            1221 Jun 16  2015 default.xml
-rw-rw-rw-  1 0       0         4121690 Jun 16  2015 mpcs.Z
-rw-rw-rw-  1 0       0          345088 Jun 16  2015 oam.db
-rw-rw-rw-  1 0       0              22 Jun 16  2015 version.txt
```

图7-39　Femtocell内部文件列表

尽管之前该设备可以从Web层面进行固件下载，但是由于设备进行自动升级修补了该漏洞，所以无法对最新的固件进行分析。

在找到重要文件的绝对地址之后，研究人员通过FTP将这些文件分别下载到本地进行研究，如图7-40所示。

```
C:\Users\Marvin>tftp 192.168.197.241 PUT test.txt
Transfer successful: 24 bytes in 1 second(s), 24 bytes/s

C:\Users\Marvin>tftp 192.168.197.241 GET test.txt
Transfer successful: 24 bytes in 1 second(s), 24 bytes/s

C:\Users\Marvin>_
```

图7-40　Femtocell Telnet服务测试图

值得一提的是，设备使用的6.8版本的VxWorks在国内是不公开发售的，通过种种努力，获得VxWorks6.8版本的开发套件之后，研究人员对重要文件的压缩算法进行分析，发现VxWork的文件系统的压缩算法采用了deflate算法，但是直接采用zlib-flate命令是无法进行解压缩的。因为VxWork在压缩后的文件头部添加了4个字节的头部magic数，紧跟着是4个字节的文件大小，最后是1个字节的标志位0x08；所以只要对这前9个字节进行截断，将剩余的文件内容另存为要使用的文件，采用Linux系统中的zlib-flate命令进行解压缩即可，如图7-41所示。

```
^Cmarvin@ubuntu:~/Work/Femto$ strings mpcs.out | grep -i "copyright"
Copyright 1984-2009  Wind River Systems, Inc.
(c) COPYRIGHT 1989-2002, Trillium Digital Systems, Inc.
(c) COPYRIGHT 1989-1998, Trillium Digital Systems, Inc.
(c) COPYRIGHT 1989-1998, Trillium Digital Systems, Inc.
(c) COPYRIGHT 1989-2000, Trillium Digital Systems, Inc.
@(#) IPIKE $Name: ipike-any-r6_8_x20120419 $ - INTERPEAK_COPYRIGHT_STRING
Copyright (c) 2000-2009, Interpeak AB <www.interpeak.com>
Copyright (c) 2000-2009, Interpeak AB <www.interpeak.com>
@(#) IPSCTP $Name: VXWORKS_NETWORKING_6_8_20130419 $ - INTERPEAK_COPYRIGHT_STRING
]]]]]]]]]]]]]]]]]]]]]]]]]]        Copyright Wind River Systems, Inc., 1984-2009
_PJP_CPP_Copyright
```

图7-41　固件解压成功验证图

从图7-41中我们可以看到，解压缩后的程序不再是乱码状态，已经可以搜索出版权字符串。这时就可以把解压缩出来的系统主要文件拖入IDA Pro进行详细逆向分析了。但是文件在系统运行时会有加载基址，需要知道文件本身的加载基址才能方便IDA进行分析。在启动日志中，研究人员发现mpcs.Z在解压缩后加载在0xc0100000地址，在IDA Pro中填入该地址即可开始正常逆向分析工作，如图7-42所示。

```
Loading /tffs0/user1/mpcs.Z...
Begin uncompressing...
entry = 0xc0100000
[VxWorks Boot]:
```

图7-42　固件加载基址

通过对文件系统中的系统登录验证程序进行分析，发现系统添加用户的方式如图7-43所示。明文密码先经过一次MD5加密，再对密文进行base64转换，研究人员因此反向推算出登录密码的MD5值，最终破解出系统的登录密码。

```
int usrSecurity()
{
  loginInit();
  loginUserAdd((int)"SYSTEM_2G", (int)"7318gRjwLFtklgfdXT+MdiMEjJwGPVMsyUxe16iYpk8=");
  return shellLoginInstall(loginPrompt2, 0);
}
```

图7-43　系统添加用户的方式

知道了用户名和密码，就可以顺利连接VxWorks的调试shell。获得系统的整体VxWorks权限之后，研究人员仔细分析了系统所有文件及流量的加密方式，发现系统在加密流量的方法上，采用了修改后的StrongSwan IPSec加密方法。在进行流量解析之前，网络流量在WireShark中如图7-44所示。

经过对系统输出日志和系统对核心网发送数据封装程序的逆向之后，研究人员逐字节对捕获到的数据包封包格式进行解析，并且依据设备的封包规则对WireShark的解析器进行定制化，从而正确解析出在设备工作范围内通过家庭基站进行的拨打电话、收发短信、网络流量等行为的数据包。在实验过程中，研究人员抓取到的上网流量数据包解析后如图7-45所示。

1	0.000000	192.168.0.35	111.206.50.34	UDP	382	Source port: 60295 Destination port: 60295
2	5.011609	192.168.0.35	111.206.50.34	UDP	382	Source port: 60295 Destination port: 60295
3	15.027760	192.168.0.35	111.206.50.34	UDP	382	Source port: 60295 Destination port: 60295
4	35.043415	192.168.0.35	111.206.50.34	UDP	382	Source port: 60295 Destination port: 60295
5	61.093004	192.168.0.35	115.181.37.74	UDP	382	Source port: 60295 Destination port: 60295
6	66.112433	192.168.0.35	115.181.37.74	UDP	382	Source port: 60295 Destination port: 60295
7	76.124345	192.168.0.35	115.181.37.74	UDP	382	Source port: 60295 Destination port: 60295
8	96.139834	192.168.0.35	115.181.37.74	UDP	382	Source port: 60295 Destination port: 60295
9	122.023849	192.168.0.35	221.179.140.118	UDP	382	Source port: 60295 Destination port: 60295
10	122.055613	221.179.140.118	192.168.0.35	UDP	350	Source port: 60295 Destination port: 60295
11	122.411172	192.168.0.35	221.179.140.118	UDP	266	Source port: 60296 Destination port: 60296
12	122.442892	221.179.140.118	192.168.0.35	UDP	194	Source port: 60296 Destination port: 60296
13	122.450840	192.168.0.35	221.179.140.118	UDP	170	Source port: 60296 Destination port: 60296
14	122.485616	221.179.140.118	192.168.0.35	UDP	114	Source port: 60296 Destination port: 60296
15	122.491044	192.168.0.35	221.179.140.118	UDP	146	Source port: 60296 Destination port: 60296
16	124.027382	221.179.140.118	192.168.0.35	UDP	134	Source port: 60296 Destination port: 60296
17	124.053016	221.179.140.118	192.168.0.35	UDP	250	Source port: 60296 Destination port: 60296

图7-44　加密流量

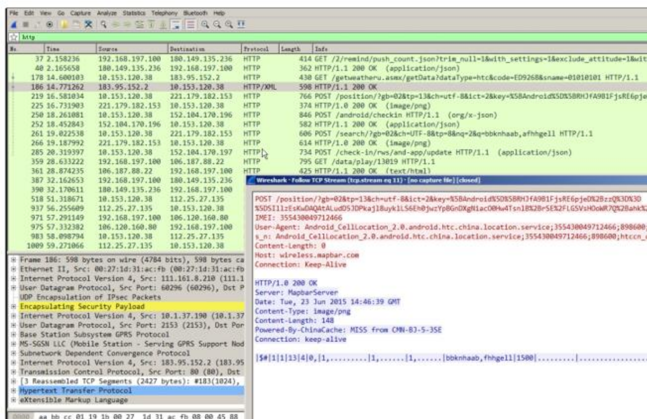


图7-45　解析后的手机上网流量数据包

由于本案例中的家庭基站的开发人员在实现协议栈时，运营商对接协议也同样由他们开发，因此并未选用标准的GSM协议封包，而是采用了自己修改定制化后的协议栈，所以对整体的分析产生了很大阻力，这也从侧面验证了“开发留心一点，破解多搞一年”这句话。

本案例中，与常见的家庭基站破解不同的是，设备的操作系统为不常见的VxWorks，并且开发人员在开发过程中未采用标准协议，给整体的破解工作带来了重重阻力。此外，常见的GSM伪基站的特征较多，可以被一些手机安全软件直接识别，但是经过修改后的此种设备，实际上是一个合法的基站，因此可轻易绕过诸多反制监测手段。

7.3.5　350元玩转Femto

在美国拉斯维加斯举行的BlackHat2015大会上，安全研究人员Alexey和Alexander另辟蹊径（Advantures in Femtoland，350yuan for invaluable fun），展示了一款从中国淘宝上用350元买到的华为飞蜂窝设备。与以上两个案例不同的是，这个飞蜂窝不仅是基于VxWorks的，而且是3G制式信号。与GSM不同的是，3G是双向鉴权机制，因此在飞蜂窝破解上有诸多难处，值得借鉴与参考。

研究人员在经过“海淘”之后，从淘宝上购买了一个350元的华为UAP21053G飞蜂窝，如图7-46所示。



图7-46　淘宝上的华为UAP2105 3G 2C蜂窝设备

与360UnicornTeam类似，这两位安全研究人员对硬件进行拆解、固件分析后，发现这同样是一款基于VxWorks的飞蜂窝设备，但是这款设备的VxWorks并未经过修补，所以在使用H.D.Moore发表的漏洞dump出完整内存之后，他们使用IDA Pro进行了进一步分析。

在对硬件分析的过程中，研究人员找到了系统的JTAG口，因为他们没有在淘宝上买J-Link，所以只好用树莓派和OpenOCD组装成一个“穷人的JTAG”，如图7-47所示。

连接配置好JTAG之后，研究人员就可以正常使用gdb来对系统进行调试了。但是这时调试依然很不方便，可以看到只能通过gdb命令一步一步对系统进行分析和调试，如图7-48所示。即使对于gdb调试经验丰富的人来说，完整分析整个系统也是一个浩大的工程，更何况

是并不精通于调试的安全分析员了。



图7-47　Femtocell调试硬件

```
(gdb) info registers
r0             0x7c     124
r1             0xafc8000        184320000
r2             0x0      0
r3             0x1914a88        26299016
r4             0x270f7ec        40957932
r5             0x270f7f0        40957936
r6             0x1914a90        26299024
r7             0x1      1
r8             0x4df00  319232
r9             0x2c092e0        46174944
r10            0x2      2
r11            0x270f814        40957972
r12            0x270f7f0        40957936
sp             0x270f7ec        0x270f7ec
lr             0x2      0x2
pc             0x1575dc 0x1575dc
cpsr           0x60000013       1610612755
(gdb) x/10i 0x1575dc
=> 0x1575dc:    ldr     r2, [pc, #452]  ; 0x1577a8
   0x1575e0:    ldr     r3, [r2]
   0x1575e4:    cmp     r3, #1
   0x1575e8:    beq     0x1576d8
   0x1575ec:    mov     r4, #0
   0x1575f0:    str     r4, [r2]
   0x1575f4:    bl      0x197c58
   0x1575f8:    ldr     r3, [pc, #428]  ; 0x1577ac
   0x1575fc:    ldr     r2, [r3]
   0x157600:    mov     r5, r4
(gdb) si
0x001575e0 in ?? ()
```

图7-48　使用gdb命令分析和调试系统

所以他们紧接着对OpenOCD的源代码进行修改，使用改动后的gdb_server对VxWorks进行调试，即可使OpenOCD支持IDA。修改代码如图7-49所示。

对OpenOCD进行重新编译加载之后，IDA Pro即可对系统进行动态调试，设置断点、反编译、C伪代码等强大的工具都可以加速整个逆向研究过程，事半功倍。如图7-50所示即为对系统进行动态调试时IDA Pro的工作截图。

在对设备本身的结构和执行流程研究清楚之后，研究人员开始从系统的整体架构入手（包含运营商网关），系统架构图如图7-51所示。

openocd-make-ida-happy.diff　Raw

```
diff --git a/src/server/gdb_server.c b/src/server/gdb_server.c
index 4a33a30..aa6ef53 100644
--- a/src/server/gdb_server.c
+++ b/src/server/gdb_server.c
@@ -2196,7 +2196,8 @@ static int gdb_get_target_description_chunk(struct target *target, struct target
        else
                transfer_type = 'l';

-       *chunk = malloc(length + 2);
+       // *chunk = malloc(length + 2);
+       *chunk = malloc(tdesc_length + 2);
        if (*chunk == NULL) {
                LOG_ERROR("Unable to allocate memory");
                return ERROR_FAIL;
@@ -2741,7 +2742,7 @@ static int gdb_input_inner(struct connection *connection)
                                         * sent by GDB first to OpenOCD, thus defeating the check to
                                         * make only the single stepping have the sync feature...
                                         */
-                                       nostep = true;
+                                       // nostep = true;
                                        LOG_WARNING("stepi ignored. GDB will now fetch the register st
                                                        "from the target.");
                                }
```

图7-49　修改代码

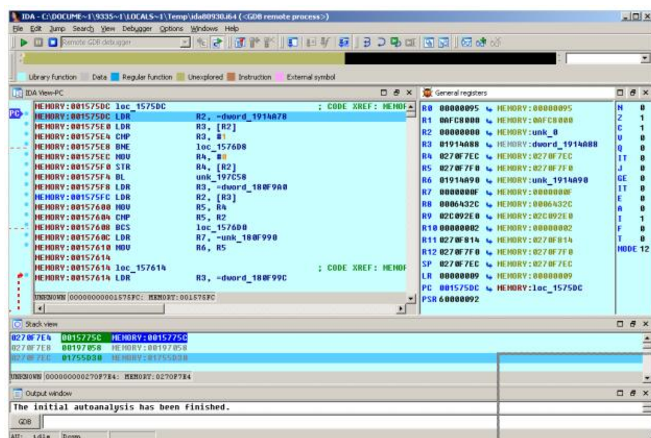


图7-50　对系统进行动态调试时IDA Pro的工作截图

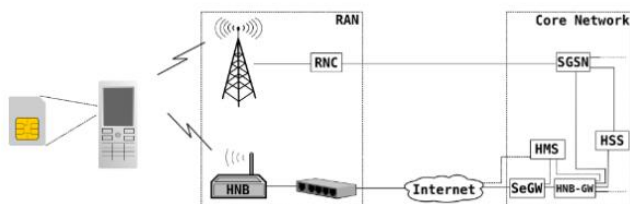


图7-51　家庭基站系统架构图

研究人员发现，在手机与基站建立完整双向鉴权之前，依然可以发送伪造的短信，包括普通短信和原始短信（闪信），而原始短信可以搜集Kc、升级SIM卡内文件系统、安装JavaCard程序。绕过验证直接发送短信的数据包如图7-52所示。

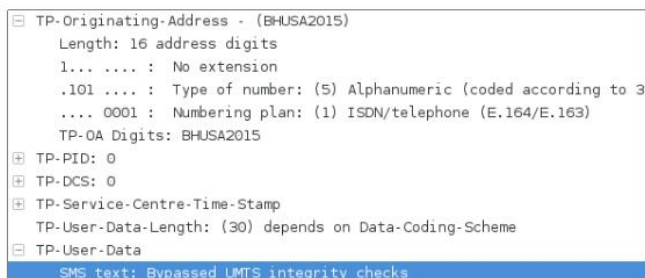


图7-52　绕过验证直接发送短信的数据包

另外一个比较重要的注意点就是，实际上在USIM支持的情况下，在UMTS环境中可以把UMTS鉴权方式降级攻击为GSM鉴权方式，从而大大简化客户端连接至假冒基站的欺骗流程。

在用户端对飞蜂窝进行鉴权时，Kc是一个重要的参数，可以通过以下途径获得这个参数值：

- 通过向SIM卡下发RAND值来获取Kc值。
- 通过原始短信获得系统内的Kc值。
- 通过接触确定的TAR，使UMTS降级为GSM验证。
- 直接用智能卡读卡器。

在飞蜂窝上，一个K_c值足够让手机相信其是一个合法的3G基站。

以上途径的攻击示意图如图7-53所示。

RANAP	124	(RUA) id-DirectTransfer (DTAP) (MM) Authentication Request
GSMTAP	69	GSM SELECT File DF.GSM
GSMTAP	83	GSM RUN GSM ALGORITHM / AUTHENTICATE [Malformed Packet]
GSMTAP	79	GSM GET RESPONSE
RANAP	108	(RUA) id-DirectTransfer (DTAP) (MM) Authentication Response
SCTP	64	SACK
SCTP	64	SACK
RANAP	144	(RUA) id-SecurityModeControl
SCTP	64	SACK
GSMTAP	69	GSM SELECT File EF.KcGPRS
GSMTAP	76	GSM UPDATE BINARY Offset=0
RANAP	112	(RUA) id-DirectTransfer (DTAP) (MM) Location Updating Accept

图7-53　3G飞蜂窝降级攻击示意图

本案例中，研究人员进一步降低了3G环境下UMTS攻击的方式，构建了一个基于合法Femtocell的3G伪基站。更进一步的是，3G无线网络被降级为2G无线网络，从而对3G网络本身所具有的更高安全性产生了新的威胁。

7.4　降级攻击

在上一节中，讲到了3G Femtocell可以从3G网络降级到2G网络，从而绕过双向鉴权的一些限制。这种攻击被称为降级攻击。

4G（LTE）网络也可以降级到2G网络。UMTS和LTE网络都有重定向（redirection）功能，设计此功能的初衷是用于把手机终端转移到其他网络中。例如当前的LTE基站负载太重，此时又有新手机连接进来，LTE基站只好把这部手机安排到其他网络中，于是它告诉这部手机，请到GSM网络的xxx频点去。这个功能被攻击者利用时，就会形成降级攻击。

例如在LTE基站上，对手机发出Attach Reject消息时，可以在RRC connection release时携带一条重定向消息。

```
RRC Connection Release
    {
      message c1: rrcConnectionRelease: {
        rrc-TransactionIdentifier 0,
        criticalExtensions c1: rrcConnectionRelease-r8: {
          releaseCause other,
          redirectedCarrierInfo geran: {
            startingARFCN 10,
            bandIndicator dcs1800,
            followingARFCNs explicitListOfARFCNs: {
              512
            }
          }
        }
      }
    }
```

以上这条消息的含义是，告诉手机在1800MHz频段，搜索ARFCN＝512频点，接入这个GSM基站，于是手机就会乖乖地落入指定的GSM陷阱中去了。

降级攻击还有一种更简单、粗暴的方式，就是直接使用大功率干扰器。干扰器可以把4G信号和3G信号全部干扰掉，只留下2G网络信号，此时手机也就别无选择，只能进入不安全的GSM网络了。但这种

方式会对覆盖半径内所有的手机起作用，影响面太大。重定向方式的优点就在于，可以精确地根据IMSI号码，对指定的手机进行攻击，只让该手机落入陷阱中。

7.5 移动通信网络中的防御措施

对于上层的应用软件来说，移动通信网络是一个管道。过去我们常常认为这个管道是比较安全的，因为基础设施控制在运营商的手中。但是，通过以上的研究我们可以看到，通过伪基站、Femto基站，攻击者可以对手机发起DoS攻击，可以窃听通话，可以截获数据流量。所以，应用软件的开发者要时刻提醒自己，这不是一个绝对安全的管道。开发者需要在应用的易用性和安全性之间做好平衡。

有很多安全研究者已经向谷歌建议，在Android系统中增加移动网络安全性的接口。例如，当GSM网络正在使用A5/0算法时，提示用户当前通话是未加密的；当手机突然从4G/3G网络降级到2G网络时，提示用户当前网络安全级别不高；每当手机经过IMSI Catcher时，都弹出提示……

如果要从根本上解决这些问题，一是需要等待若干年之后关闭GSM网络；二是需要3GPP等标准组织对3G/4G以及未来的5G系统在标准上进行改进。

第8章　卫星通信安全

黑客们对无线电的研究已经触及到卫星通信领域。例如，SatNOGS（https：//satnogs.org/）是一个DIY卫星地面站的项目。ISEE3Reboot Project（http://spacecollege.org/isee3/）这个项目利用USRP和GNU Radio，与一颗已经发射到太空60年之久的ISEE3卫星通信，试图重启它的动力系统。

在这个远距离通信场景中，也有很多的安全问题值得研究。

8.1　人造卫星概况

自从1957年10月4日苏联发射了世界上第一颗人造卫星，至今已有4000多颗人造卫星。人造卫星是发射数量最多、用途最广、发展最快的航天器。人造卫星的运动轨道取决于卫星的任务要求，按高度可以划分为低轨道（LEO）、中轨道（MEO）、高轨道（GEO）；按地球自转方向分顺行轨道和逆行轨道；另外还有一些具有特殊意义的轨道，如赤道轨道、地球同步轨道、地球静止轨道、太阳同步轨道、大椭圆轨道和极轨道。人造卫星绕地球飞行的速度快，低轨道和中、高轨道卫星一天可绕地球飞行几圈到十几圈，不受领土、领空和地理条件限制，视野广阔，能迅速与地面进行信息交换，包括地面信息的转发，也可以获取地球的大量遥感信息，一张地球资源卫星图片所遥感的面积可达几万平方千米。

GEO卫星运行于地球静止轨道，该轨道是高达三万六千公里的同步定点轨道，即在赤道平面内的圆形轨道，卫星的运行周期与地球自转一圈的时间相同，在地面上看这种卫星好似静止不动，称为同步定点卫星。该轨道卫星系统技术成熟，成本相对较低，它的特点是覆盖照射面大，3颗卫星就几乎可以覆盖地球的全部面积，可以进行24小时的全天候通信。目前可提供业务的GEO系统有INMARSAT系统、北美卫星移动系统MSAT、澳大利亚卫星移动通信系统Mobilesat。

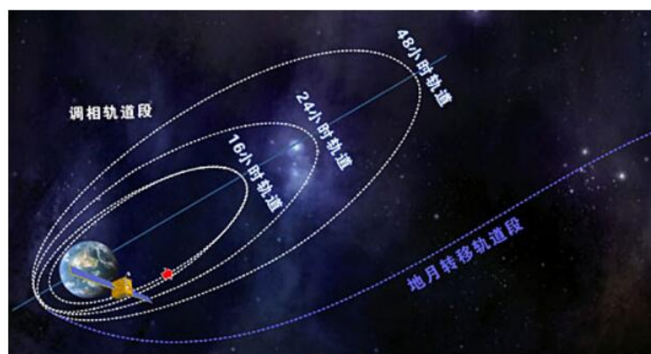


图8-1　各种类型的卫星

MEO卫星和LEO卫星相对于地面是运动的，具有传输时延短、路径损耗小、易实现全球覆盖，以及避开了静止轨道的拥挤等优点；缺

点是能够用于通信的时间短，卫星天线覆盖的区域也小，并且地面天线还必须随时跟踪卫星。目前典型的LEO系统有铱星系统Iridium、全球星系统Globalstar、Teldest等；MEO系统有奥迪赛系统Odyssey、AMSC、INMARSMT-P等。

卫星按用途可分为科学卫星、应用卫星和技术实验卫星（见图8-2）。

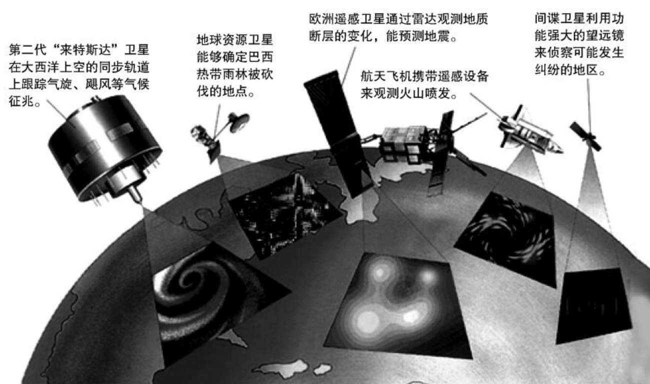


图8-2　各种人造卫星

科学卫星包含气象卫星、地球观测卫星和天文卫星等。气象卫星从遥远的太空中观测地球，不但能观测大区域天气的变化，而且针对小区域的天气变化进行观测也是其例行任务。地球观测卫星帮助科学家搜集有价值的关于地球生态系统的数据。天文卫星是对宇宙天体和其他空间物质进行科学观测的人造地球卫星。

应用卫星包含广播卫星、通信卫星、导航卫星、侦察卫星、预警卫星和反卫星卫星等。广播卫星负责直接向用户转播或发射电视广播节目。通信卫星是为了补足海底电缆通信的不足，为全球用户提供大跨度、大范围、远距离的漫游和机动、灵活的移动通信服务，是陆地蜂窝移动通信系统的扩展和延伸。导航卫星是覆盖全球的自主地理空间定位的卫星系统，允许小巧的电子接收器确定它的所在位置（经度、纬度和高度），并且经由卫星广播沿着视线方向传送的时间信号精确到10米范围内。侦查卫星主要用于对使用国家有兴趣的其他国家或地区进行情报搜集。预警卫星主要用于监视地面的情况，可以在几分钟内探测出来敌方发射导弹，根据导弹飞行弹道计算出其落点和攻击目标，并把信息上传到指挥中心，提醒做好反击准备。反卫星卫星又称自杀卫星，备有常规弹头或核弹，主要用于摧毁或干扰对方卫星。

技术实验卫星分为空间卫星和免拖拽卫星等。空间卫星主要用于空间科学研究。免拖拽卫星用于提供超静、超精、超稳的实验平台。

下面主要围绕应用卫星中的导航卫星和通信卫星进行安全研究分析。

8.2 GPS的安全研究

GPS（Global Positioning System）是由美国建造的、全世界第一个全球卫星定位系统。本节讨论GPS的安全性问题，其中大部分内容已经在2015年的DEFCON23会议上发表[1]。

8.2.1　GPS嗅探与安全分析

GPS的安全性分析已经不是一个新话题。

最著名的例子恐怕要算2011年伊朗劫持美国无人机（见图8-3）[2]。2011年12月4日，美国的一架RQ-170无人机在伊朗领空飞行，伊朗军方并没有击落它，而是使用了某种GPS欺骗的方法使这架飞机乖乖地降落在伊朗东北部的Kashmar。这架完好的无人机为伊朗军方提供了一个极好的研究标本，他们从中获取了不少技术和军事机密。



图8-3　通过GPS欺骗劫持了无人机

从那以后，对GPS欺骗的研究就越来越火了。

那么，为什么GPS接收机能够被欺骗呢？GPS卫星通过不断地广播信号，告诉接收机自己在什么位置。这是一个单向的广播信号，而且经过了从太空到地面这么远的传播，信号已经变得非常弱了。此时，如果有一个GPS模拟器，在接收机旁边假装成卫星，那么这个模拟器的信号很容易掩盖真正的GPS信号。就像图8-4所示的情况，接收机只能听到那个嗓门大的发出的信号。

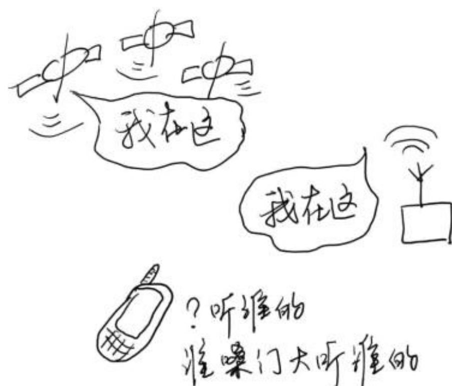


图8-4　GPS欺骗方式

这会产生哪些后果呢？美国德克萨斯州立大学奥斯汀分校Todd Humphreys教授领导的无线导航实验室[3]，是在GPS安全研究领域非常领先的一个团队。2012年，Todd Humphreys教授发表了TED演讲[4]，呼吁公众注意GPS安全性；2013年，这个团队成功欺骗了一艘游艇，改变了航向（见图8-5）[5]；2014年，他们又成功欺骗了一架无人机，控制了它的飞行位置（见图8-6）[6]。



图8-5　通过GPS欺骗改变了游艇的航向



图8-6　通过GPS欺骗控制了无人机的飞行位置

除了可以扰乱定位系统，GPS欺骗还可以扰乱定时系统，改变通信基站、金融交易系统的授时信息。据说这样的时间差，可以在交易中抢得先机，从中获利[7] 。在军事应用中，这种攻击就会产生更严重的后果。1996年，中国两枚导弹发射失败，疑因GPS被美军做了手脚[8] 。

说到这里，要介绍一下GPS民用信号和军用信号的差别。

GPS卫星在1575.42MHz发射的1.023MHz带宽的信号，是民用信号，对全球开放，标准是完全公开的[9] 。各个芯片厂家可以根据这个标准设计GPS芯片，应用在各种电子设备中。

GPS卫星在1227.60MHz发射的10.23MHz带宽的信号，是军用信号。这种信号使用了高强度的加密措施，只有美国军方能够使用。破解不是不可能，只是复杂度很高，也很难实现欺骗攻击。也就是说，除了美国军方，其他人都只能使用非常开放的民用GPS信号。

说到这里，我们再回到本章开头的例子，为什么伊朗能够劫持使用了美国“军用”GPS信号的无人机呢？有专家分析认为[10] ，伊朗可能是通过压制1227.6MHz的信号，使无人机的定位系统回退到1575.42MHz的民用信号，从而在不加密的民用信号上实现了欺骗。

8.2.2　GPS信号伪造风险评估

那么如何发射伪造的GPS信号呢？

过去做一个GPS模拟器并不是一件容易的事，GPS信号有复杂的格式、发射方法。一个商用级的GPS模拟器售价高达百万元人民币。比较简单的一种方法是“重放攻击”，也就是录制一段GPS信号，然后再播放出来，就像图8-7所示的那样。

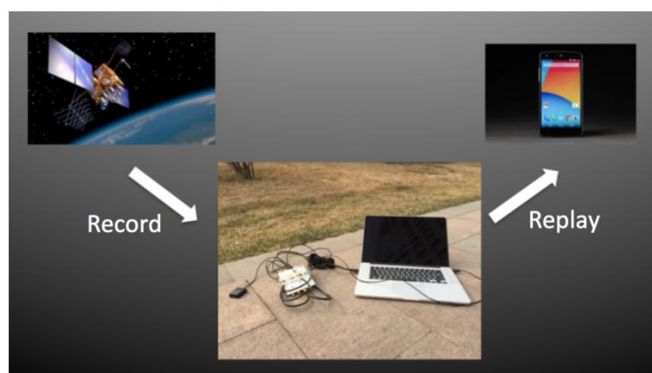


图8-7　GPS重放攻击

但是，随着软件无线电技术的不断发展，出现了越来越便宜的GPS欺骗设备，以及丰富的开源代码。现在，攻击者通过现成的开源代码，配合软件无线电设备，就能够发射GPS信号，在任意地点、任意时间，以假乱真！例如德克萨斯州大学奥斯汀分校学生基于DSP开发板制作的GPS欺骗器价格大约3000美元。然而，360UnicornTeam研究发现，目前已经可以用成本更低的通用的软件无线电设备，例如USRP/bladeRF/HackRF，以及网络上免费的开源代码，实现一个GPS欺骗器。这使得攻击成本接近于0。

1.GPS信号系统原理

GPS系统的基本原理是怎样的呢？

全球定位系统（GPS）主要由空间星座部分、地面监控部分和用

户设备部分组成。

地面监控部分主要由1个主控站、3个注入站和5个监测站组成，该部分主要用于检测和控制卫星的运行、编算卫星星历、监控系统时间。

用户设备部分主要是GPS接收机系统，用于接收并处理卫星信号来提供导航定位信息。其主要分为两类：导航型接收机（手持式、车载式、机载等）和测量型接收机（单频机和双频机）。

空间星座部分由21颗工作卫星和3颗备用卫星组成[11]。24颗卫星均匀分布在6个轨道面上，即每个轨道面上有4颗卫星。卫星的分布经过了巧妙的设计，目的是保证在地球表面和近地空间的任何一点、任何时刻至少可以观测到4～8颗卫星。

为什么至少一定要4颗卫星呢？GPS定位原理示意图如图8-8所示。

GPS定位是通过测距来实现的。GPS接收机要测量每颗卫星与它之间的距离，而距离等于光速c与时间之积，这就构成一个方程式 $c(t_s - t'_\tau)$ 。因此，距离的测量变成了时间的测量。正如图8-8所示，地平面上的GPS接收机可以接收到4颗卫星信号，假设所有卫星在T_0时刻一起发送各自的报文信息，信息通过不同的路径到达接收机。由于每条信息到达接收机的时间都带有不同程度的延迟，因此到达时间各不相同，图中 t_{s1}、t_{s2}、t_{s3}、t_{s4} 分别为4颗卫星到达接收机经过的时间。

$$\rho_1 = c(t_{s1} - t'_\tau) = \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} + cb$$

$$\rho_2 = c(t_{s2} - t'_\tau) = \sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2} + cb$$

$$\rho_3 = c(t_{s3} - t'_\tau) = \sqrt{(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2} + cb$$

$$\rho_4 = c(t_{s4} - t'_\tau) = \sqrt{(x-x_4)^2 + (y-y_4)^2 + (z-z_4)^2} + cb$$

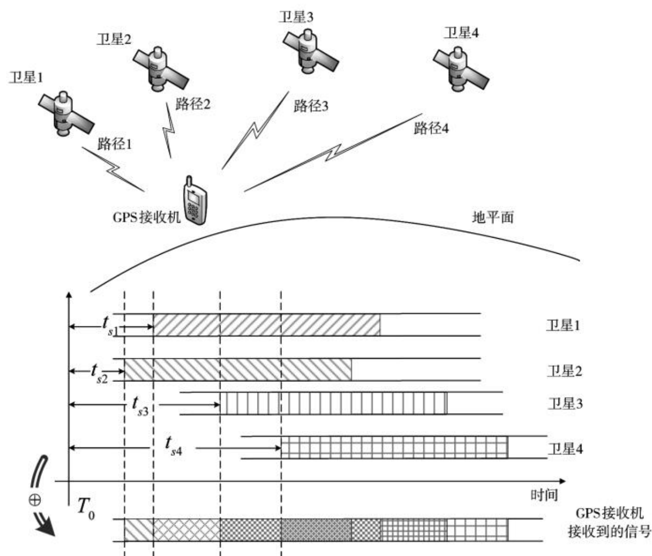


图8-8 GPS定位原理示意图

GPS系统中的卫星采用了技术复杂、成本高昂的铷或铯原子钟，相对频率稳定度可达10^{-12} ~10^{-14} 。并且地面的控制站也需要时刻监控卫星上的原子钟，在需要的时候进行校准，可以说GPS系统首先是一个严格的时间同步系统。因此，我们解析得到的卫星时钟信息是非常精准的。

而生活中GPS接收机五花八门，根据使用目的的不同，用户要求的GPS信号接收机也各有差异，我们无法为成千上万个用户的GPS接收机配置一台原子钟。因此，一台廉价的接收机的本地时钟极有可能是不准确的。而接收机时钟不准确对定位的影响有多大呢？我们知道，光1s可以走30万千米，如果某用户携带的接收机时钟错了1ms，那么带来的距离测量值就会错300km，实在是差之毫厘，谬以千里。如此一来，需要引入第4个未知数b。公式中t'_τ指的是接收机自身的时钟信息，即不准确的本地时间。这时假设 t'_τ 和精确的本地时间 t_τ 偏差为b，即 $t'_\tau = t_\tau$ -b。由于卫星的位置是已知的，而接收机的空间位置（x，y，z）未知，时间也是未知的，所以一共有4个未知数。一颗卫星可以列出一个方程，因此需要4个方程才可以解出（x，y，z，b）4个未知数。由此可知，GPS卫星一直向用户传递的是它们何“时”在何“地”的重要信息。

2.GPS信号结构

在了解了GPS定位的基本原理后，你一定很好奇想了解：卫星是怎样将如此复杂的时间信息发送到接收机的？接收机又是通过哪些信息被动定位的？

我们知道GPS星座由24颗卫星组成，且卫星不是静止的，而是动态运行的。虽然GPS卫星的位置在时时刻刻发生改变，但依据其轨道参数和参考时间就可以计算出任意时刻的精确位置，因此可以认为GPS卫星是动态已知位置的定位参考点。

GPS卫星在太空中通过1575.42MHz和1227.6MHz发送1.023MHz带宽的信号，地面用户通过手中的接收机就可以检测并处理来自太空的微弱信号，所以不得不提GPS信号结构。如图8-9所示，GPS信号主要由载波、PRN码和导航电文三者的乘积构成。

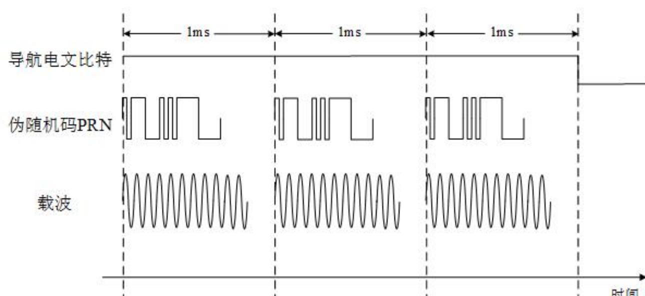


图8-9　GPS信号的基本构成示意图

民用信号的载波为1575.42MHz。

PRN码即伪随机噪声码（Pseudo Random Noise code），正是通过该PRN码的扩频特性保证了地面用户能够检测并处理微弱的信号；同时，也正是由于这些PRN码的自相关和互相关特性，保证了共享相同载频的多颗卫星信号可以彼此区分开来。

导航电文是GPS信号中调制的数据，在整个GPS信号构成中占据重要地位，每一颗卫星都在连续不断地发送其自身独特的导航电文。导航电文主要提供了信号的精确发射时间和卫星的星历数据，通过这两项数据接收机可以得到卫星的精确位置。同时，导航电文还提供了卫星的健康状态、卫星配置信息、Anti-Spoofing技术、电离层和对流层校正参数、历书数据等。以上轨道参数信息一方面用于提高定位的精确度；另一方面用于加快信号的捕获。

细心的读者通过图8-9会发现，PRN码的周期为1ms，那么导航电

文的周期又是多少呢？下面我们简要介绍一下导航电文的层次结构。

导航电文可以被分为5个不同的层次结构，如图8-10所示。最基本的结构是数据位，导航电文的数据率为50bps，即20ms输出一个有效的数据比特，可见GPS调制的数据率是相当慢的，这样做的目的是为了有足够的扩频增益，保证足够低的误码率；第二级结构是字（Word），30个数据位组成一个字；第三级结构是子帧（Subframe），10个字组成一个子帧，所以一个子帧包含300个数据比特；第四级结构是页面（Page）或帧，5个子帧组成一个页面；第五级结构是25个页面组成一个整周期的导航电文。每颗卫星都在连续不断地发射着25个页面组成的周期导航电文。导航电文格式详见参考文献[12]。

接收机接收每颗卫星的星历数据来确定卫星的位置。它还需要传输时间和钟差改正数据来计算伪距（也就是公式中的值），进而确定接收机的位置。这些信息在前三个子帧中传输，所以接收机至少需要16s（在最坏的情况下是30s）来获取这些必要信息。

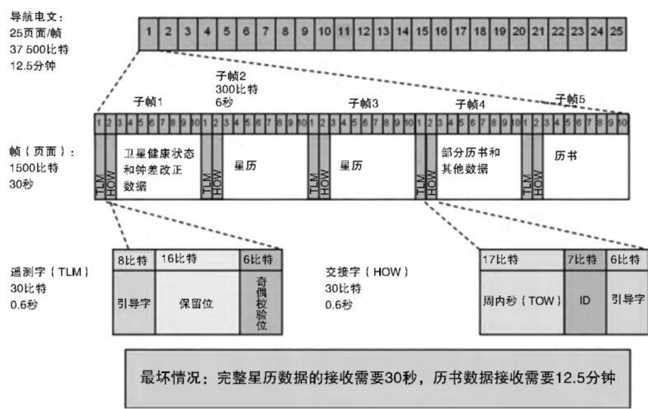


图8-10　GPS导航电文层次结构图

然而，卫星在空间传输的过程中，卫星以3～4km/s的速度高空飞行，同时用户也可能处于运动状态，所以用户接收到的卫星信号必然存在多普勒频移现象。还要考虑卫星与用户的终端存在着相对运动，即多普勒效应。下面我们先来了解一下多普勒效应。

在我们的生活中多普勒效应的应用非常广泛。例如，当一辆救护车迎面驶来时，我们听到的声音大多尖锐、响亮，而在离去时声音会越来越微弱。在医学中，可以通过测量超声波的反射波波长变化来得到彩色的血流图像，即彩超。而多普勒效应不仅适用于声波，而且也

适用于所有类型的波，包括电磁波。例如，警车可以给某辆车发送雷达信号，通过测量发射回来的雷达波频率的变化而得知某辆车的车速，即雷达测速仪。同样，由于光的波粒二象性，光波也存在多普勒效应。在天文学中，多普勒效应可以测量恒星相对于我们的径向速度。这主要是由于恒星光谱会显示出离散的频率和波长，通过测量这些光波的变化就可以知道恒星是在远离我们还是在靠近我们。我们知道红色的波长比蓝色的波长长，因此向着波源运动时，波被压缩，波长变短，频率变高，称为蓝移；相反，背离光源运动时，波长会相应变长，与之对应称为红移。所以，当以足够快的速度通过路口红绿灯时，是极有可能将红灯看成绿灯的，这是由于发生了蓝移，波长传到我们眼中变短所致。

在GPS接收机内部，获取多普勒频移的方法往往有两种。

一种方法是利用载波相位观测量。假设t时刻的载波相位为φ_t，t＋1时刻的载波相位为φ_{t+1}，那么多普勒频移为：

$$f_d = \varphi_{t+1} - \varphi_t$$

该方法主要是基于载波频率是相位的导数得到的，因此该方法具有高精确性。但该方法的缺点是必须保证载波相位没有发生周跳，一旦发生周跳，多普勒频移观测量也会有一个很大的跳变。这种方法对于高精度测绘用接收机比较实用，因为高精度测绘接收机的应用环境一般都在开阔地带，所以GPS信号一旦锁定，就能保持较长时间无周跳的载波相位观测量输出。而对于一般低成本的单频接收机，比如车载或手持导航用接收机，由于其应用环境的多变，所以经常发生信号失锁，于是其载波相位观测量也会经常发生周跳，故这种方法就不再适用。

另一种方法是直接读取载波NCO的配置值，将该值转化成频率后与射频前端的标称中频值f_c相比，其差就是多普勒频移观测量。关于载波NCO的工作原理和标称中频值这里不作详述，有兴趣的读者可以参考文献[13]。

3.GPS模拟器的实现过程

在清楚了GPS信号的构成原理后，再来看看Unicorn Team是如何零成本实现这个复杂的过程的。

（1）GNSS软件无线电接收机

下面先来了解一下如何解调室外的GPS信号。这是为了给GPS模拟器做一个调试和验证工具。

首先需要搭建一个GNSS软件无线电接收机。采用USRP B210无线电设备，其GNSS软件无线电接收机在GNU Radio框架下模块的连接框图如图8-11所示。

注：GNSS是所有在轨工作的卫星导航系统的总称呼，目前主要包括GPS（全球定位系统）、GLONASS（全球导航卫星系统）、北斗卫星导航系统、WAAS（广域增强系统）、EGNOS（欧洲静地卫星导航重叠系统）、DORIS（星载多普勒无线电定规定位系统）、PRARE（精确距离及其变率测量系统）、QZSS（准天顶卫星系统）、GAGAN（GPS静地卫星增强系统），以及正在建设的Galileo（卫星导航定位系统）、中国的GNSS导航定位系统和IRNSS（印度区域导航卫星系统）。

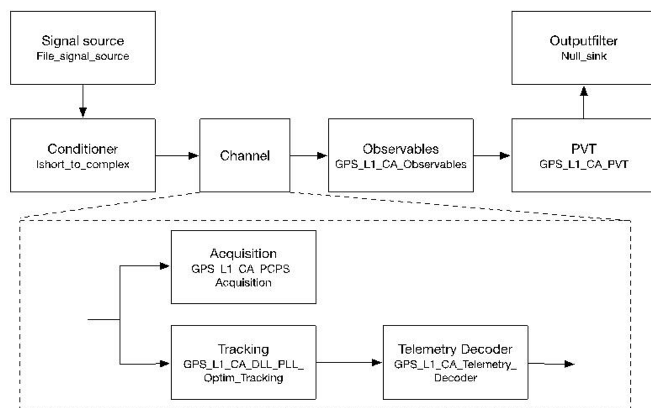


图8-11　GNSS软件无线电接收机模块连接框图

GNSS-SDR软件无线电接收机成功检测到室外GPS信号的关键是（见图8-12）：

- 低噪声放大器（LNA）。本实验采用了Mini-Circuit ZX60-V82-S＋。

- GPS Active天线，需要Bias-Tee电路对天线供电。本实验采用了Mini-Circuit ZX85-12G-S＋。



图8-12　带LNA和GPS Active天线的接收机

其中，Active天线增益不详；ZX60的标称增益大约是14dB[14]；B210设置的RX增益是60dB。

然后对户外采集到的GPS信号进行线下解调。在这里，采用GNSS-SDR程序[15]。修改压缩包中的.conf文件后，运行可以得到如下结果：

```
NAV Message: received subframe 3 from satellite GPS PRN 20 (Block IIR)
Ephemeris record has arrived from SAT ID 20 (Block IIR)
NAV Message: received subframe 3 from satellite GPS PRN 1 (Block IIF)
Ephemeris record has arrived from SAT ID 1 (Block IIF)
NAV Message: received subframe 3 from satellite GPS PRN 32 (Block IIA)
Ephemeris record has arrived from SAT ID 32 (Block IIA)
NAV Message: received subframe 3 from satellite GPS PRN 11 (Block IIR)
Ephemeris record has arrived from SAT ID 11 (Block IIR)
NAV Message: received subframe 3 from satellite GPS PRN 17 (Block IIR-M)
Ephemeris record has arrived from SAT ID 17 (Block IIR-M)
(new)Position at Lat = 41.27470287549986 [deg], Long = 1.987520691432634 [deg],
(new)Position at Lat = 41.27476474579162 [deg], Long = 1.987698559529529 [deg],
(new)Position at Lat = 41.2747702419555 [deg], Long = 1.987607056533979 [deg], H
(new)Position at Lat = 41.27474204858431 [deg], Long = 1.987590043994324 [deg],
(new)Position at Lat = 41.27474697031863 [deg], Long = 1.987670036318851 [deg],
```

可以看出，线下解出的地理位置为：纬度41.27474，经度1.987607，海拔高度70m左右，与我们测试的位置一致。如此一来，GNSS-SDR软件无线电接收机的搭建就大功告成了，这样我们就有了GPS信号软件无线电发射机的验证工具。

（2）GPS信号发射机

GPS信号软件无线电发射机采用黄琳和贾立伟共同完成的GPS模拟器。其MATLAB程序详见参考文献[16]，程序主要功能就是根据星历数据文件按照导航电文的格式填充对应位信息，同时考虑到地球的自传和卫星距离的遥远，因此需要计算对应的多普勒效应。MATLAB程序生成导航电文数据后，通过USRP或bladeRF等软件无线电设备发射出去即可。

在MATLAB程序中，我们挑选了仰角最高的5颗星，产生5个通道的数据，再根据它们各自的伪距合并在一起。

```
-----------------
Time: 2015-2-14 8:30:0
N 39.9813
E 116.4841
H 100
-----------------
Satalite 8 telegraph for 1th channel generating...
Satalite 14 telegraph for 2th channel generating...
Satalite 25 telegraph for 3th channel generating...
Satalite 31 telegraph for 4th channel generating...
Satalite 32 telegraph for 5th channel generating...
Total 5 satelite telegraphs are generated
All 5 available channels are filled.
Page 1 generating...
Page 2 generating...
Page 3 generating...
Page 4 generating...
Page 5 generating...
Page 6 generating...
Page 7 generating...
Page 8 generating...
```

上面是MATLAB程序的输出信息。设定的地理位置是：纬度39.9813，经度116.4841，海拔100m。PRN8，14，25，31，32，一共产生了5颗星的信号。总共产生了8个页面的信号，也就是长度为240s（6min）的数据。

如图8-13所示是GNSS-SDR离线解调得到的输出。正确地收到了5颗星的星历数据，定位结果也与MATLAB设定的一致。

```
test@ub1404: ~/gnss-sdr/install
Height= 84.96576727088541 [m]

NAV Message: received subframe 3 from satellite GPS PRN 14 (Block IIR)
NAV Message: received subframe 3 from satellite GPS PRN 31 (Block IIR-M)
Ephemeris record has arrived from SAT ID 14 (Block IIR)
Ephemeris record has arrived from SAT ID 31 (Block IIR-M)
Current input signal time = 228 [s]
NAV Message: received subframe 3 from satellite GPS PRN 25 (Block IIF)
Ephemeris record has arrived from SAT ID 25 (Block IIF)
NAV Message: received subframe 3 from satellite GPS PRN 32 (Block IIA)
Ephemeris record has arrived from SAT ID 32 (Block IIA)
NAV Message: received subframe 3 from satellite GPS PRN 8 (Block IIA)
Ephemeris record has arrived from SAT ID 8 (Block IIA)
(new)Position at Lat = 39.98136919351661 [deg], Long = 116.4842187915581 [deg],
Height= 12.47768028080463 [m]

(new)Position at Lat = 39.98126111361005 [deg], Long = 116.4842753681772 [deg],
Height= 99.35894597321749 [m]

Position at 2015-Feb-14 08:33:47 is Lat = 39.98126111361005 [deg], Long = 116.48
42753681772 [deg], Height= 99.35894597321749 [m]
(new)Position at Lat = 39.98047187558485 [deg], Long = 116.4842925131961 [deg],
Height= 89.90765669662505 [m]

(new)Position at Lat = 39.9819037778793 [deg], Long = 116.4838705542527 [deg], H
eight= 101.0470515359193 [m]

(new)Position at Lat = 39.98190471292815 [deg], Long = 116.4835118857243 [deg],
```

图8-13　GNSS-SDR解调结果

接下来将通过GNSS-SDR验证后的数据用bladeRF发射出去，使用USRP B210进行接收，如图8-14所示。将接收到的卫星信号以文件格式保存。

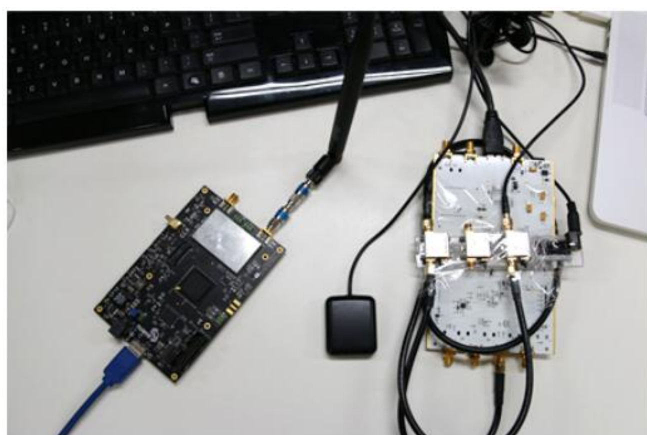


图8-14　回环测试

随后，使用Google Earth打开PVT.kml文件，通过图8-15可以看到之前设定的位置。

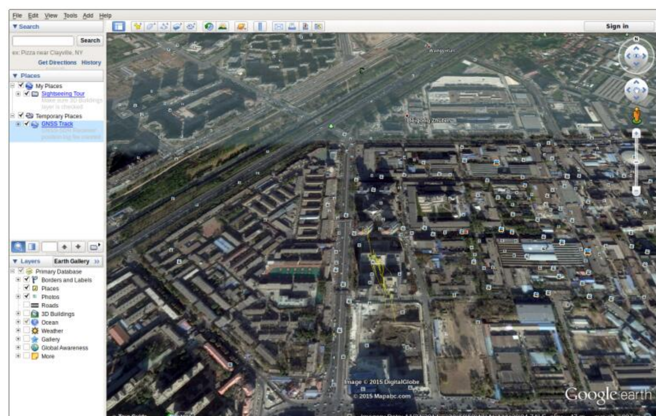


图8-15　通过Google Earth看到解析后的位置

（3）测试GPS欺骗的效果

前文提到接收机会选择信号强的GPS信号源，主要是由于该信号处在同一载波频率上，且民用信号没有进行加密，故信号强的伪GPS信号会干扰真实的GPS信号，并且劫持接收机，最后使接收机产生误判。下面我们来看看Unicorn Team自制的GPS模拟器对周围设备的影响。

- 对手机和导航设备的影响

图8-16展示了对三星Note3、Nexus5和iPhone手机GPS定位的影响，可以明显地看出，手机的位置都可以被成功欺骗到纳木错。该款软件名称为GPS Test Plus。当然可以被欺骗的手机不止这两个品牌，几乎所有的手机都无法幸免于强烈的GPS信号。

图8-17展示了对汽车GPS导航系统的欺骗，可以看出，汽车导航也被成功欺骗至纳木错。看来汽车导航也十分不可靠呀！如果你在异地迷了路，最有效的办法还是咨询当地土著居民。

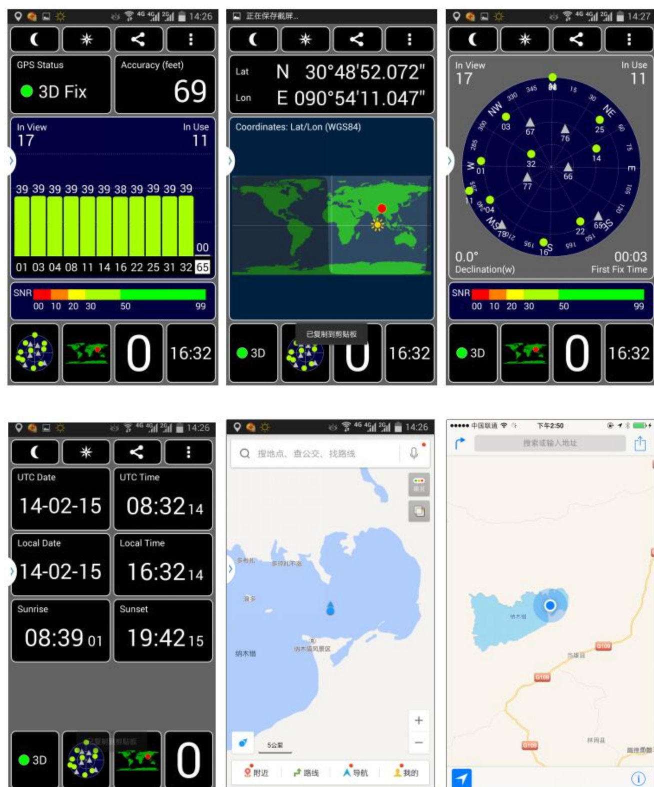


图8-16　GPS模拟器对手机定位的影响



图8-17　GPS模拟器对汽车导航定位的影响

❍ 对双模定位芯片的测试

目前市场上有些设备使用了双模定位芯片，即同时采用GPS与北

斗系统进行定位。我们在天台测试了一款使用多模芯片的儿童手表。如图8-18所示，手表定位前1分钟在上海陆家嘴，随后又变换至北京世界公园。

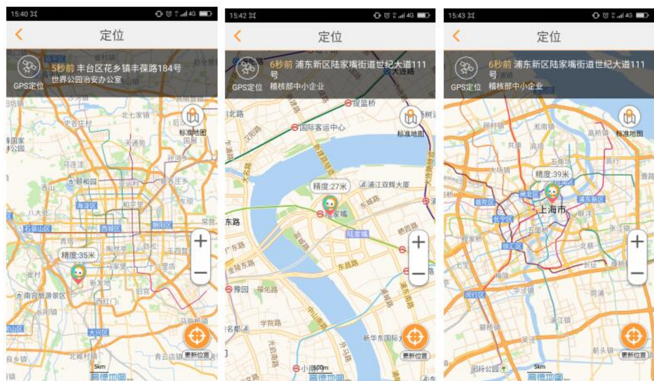


图8-18　GPS模拟器对多重定位的影响

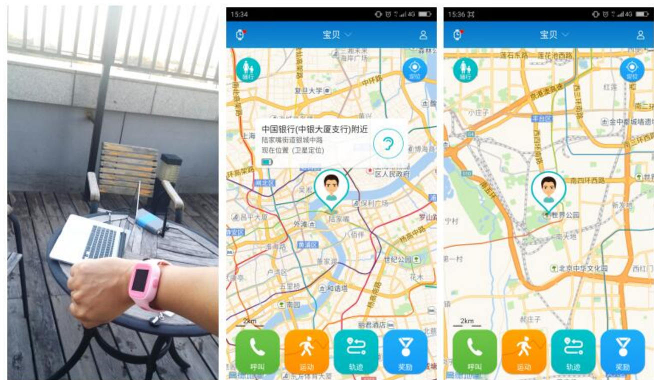


图8-18　GPS模拟器对多重定位的影响（续）

为什么北斗系统也被骗过了呢？原因是这样的：

GPS模拟器发射的伪造信号的强度非常大，各种卫星定位系统的频点基本上都是以1575.4MHz为中心设计的，而北斗的频点是在1561.098MHz，距离并不远。所以伪造信号干扰了北斗信号，使北斗接收机无法定位。因此，GPS接收机只能听到伪造的GPS信号，被定位到了伪造的位置。

- 时间欺骗测试

图8-19展示了GPS模拟器扰乱定时系统。可以明显地看出，右上角的时间为17：02，而被GPS模拟器扰乱后的时间为23：00。

❍ 对无人机的影响

接下来，让我们再看看生活中的无人机是如何被劫持的。无人机作为一种“交通”工具，在运动的时候，要能像鸽子一样记住来的地方和去的地方，实现无人操控，就需要通过GNSS进行定位。比如图8-20所示的无人机，就是通过GPS进行定位的。在室内只能人工操作，但是在户外，在遥控器上接上手机，就能很清楚地看到GPS定位出的飞行器位置，然后设置巡航地点和返航点，这样就能在无人操控的情况下进行自动巡航和自动返航。



图8-19　GPS模拟器对定时系统的影响



图8-20　户外无人机控制

无人机的发展迅速，但是带来的安全隐患却不少。由于无人机可航拍、可携带一定重量的物品，倘若无人机在任意军事基地或政治场所起飞，势必会给国家的安全带来非常大的威胁。因此，目前大多数正规品牌的无人机都开始强制进行了固件升级，设置了N多禁飞区，比如各大城市、机场都设置了禁飞区。当无人机的GPS定位到目前处

于禁飞区时，比如北京六环以内，无人机将无法启动，即使在GPS信号不好的地方起飞了，但是只要GPS信号稳定下来后检测到自己处在禁飞区内，无人机也将立即迫降；或者在禁飞区外飞到禁飞区边缘，也将自动按照失控模式进行返航或者降落处理。于是各大无人机的爱好者悲剧了，特别是处在禁飞区的各位航拍侠，会发现手机、平板上出现一个大红X，如图8-21所示，语音提示您在禁飞区，然后无人机根本不受操控，电机无法启动，即使在室内启动了带到外面，也会停掉。这时Unicorn Team自制的GPS模拟器就开始大显神通了。在无人机周围随便发射一个非禁飞区的坐标，飞机就可以正常起飞了。



图8-21　禁飞区提示

相反，假如飞机正在非禁飞区翱翔，此时通过GPS模拟器发射一个禁飞区信号会发生什么呢？看到图8-22，你肯定明白了，无人机将会直接降落或者返航。直接降落的可能性更大。就这样，成功地劫持了一架无人机。



图8-22　无人机劫持

8.2.3　防御方法及建议

目前中国仍然有大量的设备还在使用GPS定位，大到通信基站、船舶、飞机，小到手机和各种物联网设备。GPS欺骗攻击的门槛已经变得如此之低，不得不让人担忧这些系统的安全性。如今，在网上只需要两三百元就可以买到定位跟踪器，个人的位置信息泄露变得廉价就会使不法分子有机可乘。当然，有一部分人想获取他人的位置是为了安全起见，然而，位置信息的泄露也是对个人安全最严重的威胁，因此很难用法律来保障个人位置信息不会被不法分子所利用。同样的，当GPS欺骗技术也以如此廉价的形式存在于我们生活中时，谁又能保证所有的使用者都会恪守社会道德准则呢？由此，我们在三个方向上提出了保障定位信息防欺骗的技术建议。

在无法改变GPS芯片的情况下，建议开发者在应用层结合蜂窝网络定位、WiFi定位等各种信息，做一个综合的判断，避免定位定时错误可能带来的后果；假如定位芯片为多模定位芯片，还可以结合芯片所支持的其他定位系统，例如“伽利略GALILEO”、“北斗”等系统，做一个多模定位的判断，避免采用单模定位带来的脆弱性。

在GPS接收端的技术上还可以参考Todd Humphreys团队的一些研究方法[17]，例如通过干扰噪声传感器的防御方法；基于SSSC或NMA在WAAS信号上的防御；多系统多频防御；单天线防御[18]；基于扩频通信安全码在L1C频段上的防御[19]；基于L1C、L2C或L5频段上导航信息的身份验证的防御[20]；相关配置文件异常的防御[21]，以及多天线防御和基于军用信号的互交织防御办法[22]等。

除此之外，还可以在民用GPS信号的发射端进行技术改良，在可扩展的星历报文上增加数字签名，只有通过数字签名验证的报文接收机才认可其真实性。如此一来，就在根本上保证了信息的安全性。

如今，全球正在投入使用的4个卫星导航系统分别是：美国GPS系统、欧洲伽利略GALILEO系统、俄罗斯的格洛纳斯GLONASS系统和中国的北斗系统。中国经过十几年的努力，终于摆脱了对美国GPS系统的依赖，这也是中国军事实力提升的重要标志。

那么，是不是中国使用了北斗系统就可以高枕无忧了呢？不是。北斗的民用级信号同样是薄弱的，同样有可能被“重放攻击”和“欺骗攻

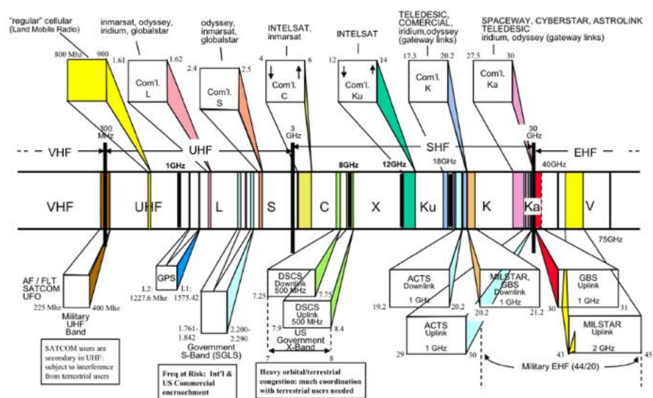
击”。360Unicorn Team也已验证了重放攻击的可行性。而且北斗民用信号的标准也是公开的，所以制作一个北斗欺骗器并不困难。因此，建议国家有关部门考虑北斗导航系统的分级授权机制，对军用系统、重要的民用系统使用不同级别的授权加密信号；最低级别的民用信号仅可用于消费电子产品；在安全性和产品成本之间取得恰当的平衡。

8.3　Globalstar系统的安全分析

除了GPS通信的民用频段不加密外，是否还会有其他卫星通信也不加密呢？如果你看过2015年BlackHat大会上Colby Moore的议题Spread Spectrum Satcom Hacking[23]就一定知道，不加密的频段不止GPS，Globalstar系统的上行链路1610～1626.5MHz同样是不加密的。

Colby Moore的议题主要是基于Globalstar系统，该系统设计简单，既没有星际电路，也没有星上处理和星上交换功能，仅仅定位为地面蜂窝系统的延伸，从而扩大了地面移动通信系统的覆盖，因此降低了系统投资，也减少了技术风险。GIobalstar系统由48颗卫星组成，均匀分布在8个轨道面上，轨道高度为1389km。它有4个主要特点：一是系统设计简单，可降低卫星成本和通信费用；二是移动用户可利用多径和多颗卫星的双重分集接收，提高接收质量；三是频谱利用率高；四是地面关口站数量较多。

在通信领域中，按照电磁波的频率范围分成几个不同的频段。图8-23中涉及了甚高频（VHF）、特高频（UHF）、超高频（SHF）和极高频（EHF）频段。其中全球星系统馈线链路使用超高频中的C频段，关口站到卫星上行链路使用5091～5250MHz，卫星到关口站下行链路使用6875～7055MHz。全球星系统用户链路使用特高频中的L、S频段，用户终端到卫星上行链路使用1610～1626.5MHz，卫星到用户终端下行链路使用2483.5～2500MHz。卫星L频段和S频段天线均由16个波束组成。L频段和S频段的频道间隔为1.23MHz。



"regular" cellular
(Land Mobile Radio)
inmarsat, odyssey,
iridium, globalstar
odyssey,
inmarsat,
globalstar
INTELSAT,
inmarsat
INTELSAT
TELEDESIC,
COMERCIAL,
iridium,odyssey
(gateway links)
SPACEWAY, CYBERSTAR, ASTROLINK
TELEDESIC
iridium, odyssey (gateway links)
Com'l. L
Com'l. S
Com'l. C
Com'l. Ku
Com'l. K
Com'l. Ka
VHF
UHF
SHF
EHF
VHF
UHF
L
S
C
X
Ku
K
Ka
V
1GHz
8GHz
12GHz
18GHz
40GHz
75GHz
AF / FLT
SATCOM
UFO
Military
UHF
Band
GPS
Government
S-Band (SGLS)
DSCS
Downlink
500 MHz
DSCS
Uplink
500 MHz
US
Government
X-Band
ACTS
Downlink
1 GHz
ACTS
Uplink
1 GHz
MILSTAR,
GBS
Downlink
1 GHz
GBS
Uplink
1 GHz
MILSTAR
Uplink
2 GHz
Military EHF (44/20)
SATCOM users are secondary in UHF: subject to interference from terrestrial users
Freq at Risk: Int'l & US Commercial encroachment
Heavy orbital/terrestrial congestion: much coordination with terrestrial users needed

图8-23　卫星频段

8.3.1　Globalstar的码分多址技术

Globalstar系统采用的是码分多址（CDMA）通信方式，相同的一组频率在每颗卫星的16个点波束中再用。在每一个频分子信道中，采用不同的伪随机码（PN）来区别不同的逻辑信道。

码分多址系统的特点之一是扩频，也就是说，用于传输信息的信号带宽远大于信息带宽；特点之二是在扩频的基础上，不论采用什么途径扩频，基本上都是用一组优选的扩频码进行控制。或者说，CDMA是在信号的编码维度上对无线信号空间的划分。顾名思义，码分多址就是给每个用户分配一个唯一的扩频码（或称地址码），通过该扩频码的不同来识别用户。对扩频码的选择要求比较苛刻，即自相关性很强，而互相关性很弱；出于系统容量的考虑，对于特定长度的地址码集还要求其能够提供足够多的地址码；在统计特性上要求地址码类似白噪声以增强隐蔽性，这在军事通信中尤为重要；为了提高处理增益，应选择周期足够长的地址码；而为了便于实现，则应选择产生与捕获容易以及同步建立时间较短的地址码。通常人们的选择就是各种伪随机码。通过图8-24和图8-25可以看出，编码与解码采用的PN码一致。

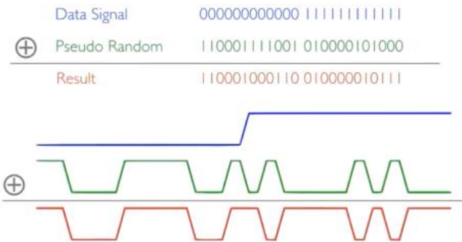


图8-24　PN码编码过程

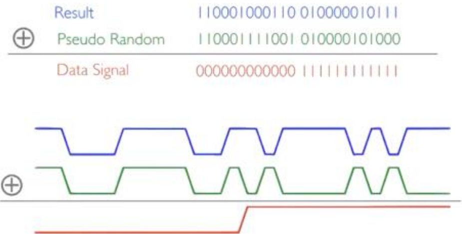


图 8-25　PN码解码过程

在了解了CDMA扩频码的产生后，让我们来看看接收端是如何恢复数据信息的。

首先，当同步系统完成信号捕获过程后，同步系统转入跟踪状态。所谓跟踪，是指使本地码的相位一直随接收到的伪随机码的相位改变，与接收到的伪随机码保持较精确的同步。跟踪环路不断校正本地序列的时钟相位，使本地序列的相位变化与接收信号的相位变化保持一致，实现对接收信号的相位锁定，使同步误差尽可能减小，正常接收扩频信号。跟踪是闭环运行的，当两端相位出现差别后，环路能根据误差大小自动调整，减小误差，因此同步系统大多采用锁相技术。

当接收机完成系统跟踪后，接下来就要开始捕获PN码序列了。

PN码序列捕获是指接收机在开始接收扩频信号时，选择并调整接收机的本地扩频PN码序列相位，使它与发送的扩频PN码序列相位基本一致，即接收机捕获发送的扩频PN码序列相位，也称为扩频PN码序列的初始同步。在CDMA系统接收端，一般解扩过程都在载波同步前进行，实现捕获大多采用非相干检测。接收到扩频信号后，经射频宽带滤波放大及载波解调后，分别送往2N扩频PN码序列相关处理解扩器，N是扩频PN码序列长。2N输出中哪个输出最大，该输出对应的相关处理解扩器所用的扩频PN码序列相位状态就是发送的扩频信号的扩频PN码序列相位，从而完成扩频PN码序列捕获。捕获的方法有多种，如滑动相干法、序贯估值法及匹配滤波器法等。

滑动相关法是最常用的方法。接收系统在搜索同步时，它的码序列发生器以与发射机码序列发生器不同的速率工作，致使这两个码序列在相位上互相滑动，只有在达到一致点时才停下来，因此称之为滑动相关法。

接收信号与本地PN码相乘后积分，求出它们的互相关值，然后与门限检测器的某一门限值比较，判断是否已捕获到有用信号。它利用了PN码序列的相关性。当两个相同的码序列相位一致时，其相关值输出最大。一旦确认捕获完成，则捕获指示信号的同步脉冲控制搜索控制钟，调整PN码发生器产生的PN码重复频率和相位，使之与收到的信号保持同步。

由于滑动相关器对两个PN码序列按顺序比较相关，所以该方法又称为顺序搜索法。滑动相关器简单，应用很广，缺点是当两个PN码的

时间差或相位差过大时，相对滑动速度过慢，导致搜索时间过长，特别是对长PN码的捕获时间过长，必须采取措施限定捕获范围，加快捕获时间，改善其性能。

经过以上的讲解，你一定觉得PN码序列的随机性使得该序列具有一定的加密性，并且它完全满足了Golomb总结的三个随机性假设：

（1）若序列的周期L为偶数，则0的个数和1的个数相等；若L为奇数，则0的个数比1的个数多1或少1。（表示序列中0和1出现的概率基本相同）

（2）在序列的一个周期内，长为i的游程占游程总数的 $1/2^i\ (i=1,2,\cdots)$ ，且在等长的游程中0的游程个数和1的游程个数相等。（表示0与1在序列中每个位置上出现的概率相同）

（3）序列的异相自相关函数是一个常数，即序列为二值自相关序列。

但是，1969年Massey发表的“移位寄存器综合与BCH译码”一文，引发了对序列研究方向的根本性变革。Berlekamp-Massey算法（简称B-M算法）指出：如果序列的线性复杂度为n，则只需要2n个连续比特就可以恢复出全部的序列。从这个结论可以看出，PN码序列的线性复杂度太小，不能够直接用来做流密码系统的密码流序列。线性反馈移位寄存器的综合问题就在于根据序列的少量比特求出整个序列所满足的线性递推关系。因此，Globalstar采用扩频通信的安全性是非常低的，黑客可以轻而易举地破解得到该信息。还可以看出，仅仅靠Golomb的三个随机性假设来评测序列是不够的，还需要一些其他的指标。下面让我们看一下如何窃取卫星通信中的上行链路信息。

8.3.2 Globalstar数据破解

全球星的数据通信过程如图8-26所示，陆地用户（用图中携带信息的车辆表示）通过GPS卫星可以获取自己的地理位置信息，随后将其信息采用码分多址的方式通过上行链路1610～1626.5MHz传输至Globalsar卫星，卫星采用弯管技术，即卫星既不解调也不解码，而是直接将数据包转发给地面卫星通信站。随后地面卫星通信站解调后将数据通过互联网以HTTP或FTP协议进行转发。除了GPS欺骗的危险，该通信系统中依然存在两个安全隐患，其中一个安全隐患是陆地用户通过上行链路将数据发送给卫星，该阶段采用扩频通信技术，扩频技术采用了薄弱的PN码序列作为密码流，通过上一节对扩频通信的了解，可以得知该密码流非常容易被黑客破解；另一个安全隐患是地面卫星通信站转发数据所使用的HTTP与FTP协议，没有在本地和远程主机之间建立加密频道，因此非常容易受到黑客的攻击，对数据进行窃取和篡改。

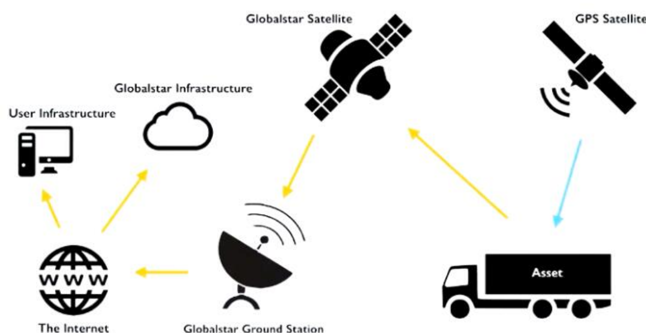


图8-26　全球星的数据通信过程

下面再介绍一款应用Globalstar进行定位的防丢神器—Spot Trace，如图8-27所示。这款防丢神器采用的是图8-29所示的通信方式，与目前市场上比较火的儿童手表的区别在于其位置信息传输至用户手机上的方式不同。无论是儿童手表还是手机均需要一张SIM卡，也就是说，该位置信息的传输方式采用的是当地运营商的网络，因此去不同的国家需要开通不同国家的通信服务；而Spot Trace采用的通信方式是卫星通信，应用无国界，当目标运动超过一定范围后会有邮件和短信提醒。每款Spot Trace都对应一个手机App应用，用户首先

需要打开蓝牙通过App与设备进行绑定，发送测试信息设备有反应即为绑定成功，关闭蓝牙即可。后续则采用Globalstar卫星进行通信，用户可以在手机联网的情况下随时随地地观测到Spot Trace的实时位置以及移动轨迹。



图8-27　Spot Trace

1.采集Globalstar的上行信号

要破解Globalstar的数据，第一步是收集信号。选择一个相对较高的楼群，如图8-28所示，该位置一方面是为了提高信号捕获的概率，另一方面是离发射源不远，降低了电磁波的各种损耗。捕获信号采用的硬件设备按照图8-29所示的方式进行搭建，整个解调设备放置在外壳中，天线则放置在外壳的外面。

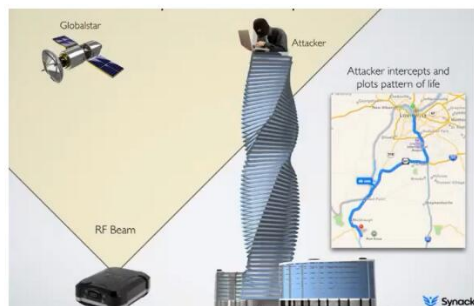


图8-28　信号捕获



图8-29　上行链路信号捕获硬件设备

所需硬件设备有：USRP B200、左手圆极化天线、通用DC-DC变换器AnyVolt3、低噪声放大器Minicircuits zx60-1614LN-S、两根SMA线、12V AC/DC适配器。

其中USRP采用GRC做出流图，如图8-30所示，用于在电脑上进行信号分析与解码。

信号采样频率设为4MS/s；信号中心频率为1.61125Hz。

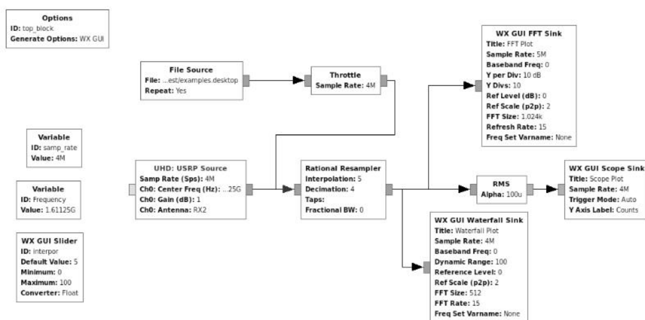


图8-30　上行链路信号捕获流图

2.破解PN码、解扩

第二步是解出信号所用的PN码，实际上是“解调”，把PSK符号转换成比特，流图如图8-31所示。这里直接调用PSK的解码模块。我们已知的PN码信息有：PN码长度255bit、采样速率5MS/s、码片速率1.25Mcps。

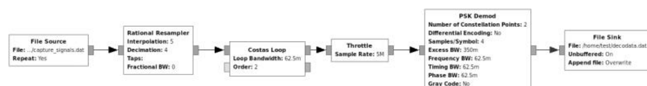


图8-31　PN码解码流图

第三步是对信号进行解扩。将上一步得到的PN码与原始信号进行相关运算。图8-32显示的是与PN码做相关之后，得到的尖锐的相关峰。解扩以后，就得到了最终需要的比特数据。下面我们可以通过该信息的数据包格式得到对应的信息内容。

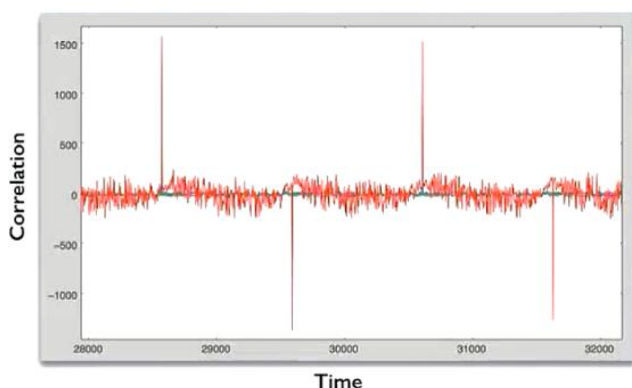


图8-32　解扩时原始信号与PN序列相关运算产生的相关峰

3.分析数据包内容

第四步是分析数据包内容，获取重要信息。数据包格式如图8-33所示，总共144比特，其中前10比特为前导码，26比特的ESN码是什么呢？这可是关键信息，它就是产品的ID号，即厂家默认为一款产品只分配一个ID注册码，我们先把这个码记录下来。中间的72比特就是数据包所传递的地理位置信息，即经纬度数据内容。后24比特是CRC校验码。看来我们获取了不少信息，下面看看可以用这些信息进行什么攻击！

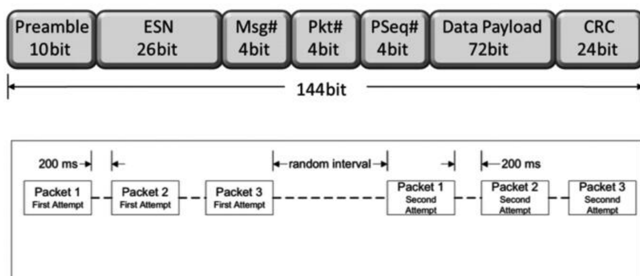

Preamble
10bit
ESN
26bit
Msg#
4bit
Pkt#
4bit
PSeq#
4bit
Data Payload
72bit
CRC
24bit
144bit
200 ms
random interval
200 ms
Packet 1
First Attempt
Packet 2
First Attempt
Packet 3
First Attempt
Packet 1
Second
Attempt
Packet 2
Second
Attempt
Packet 3
Seconnd
Attempt

图8-33　数据包格式

8.3.3 可能的攻击手法

1.克隆追踪器

Globalstar为每个Spot Trace设备都提供了更新设备固件的服务，从Globalstar的官方网站可以下载一个更新固件的应用程序，名为Spot Device Updater。这个应用程序是由SPOT3FimwareTool.jar文件编译完成的，现在我们用软件将这个文件打开，可以找到一个SPOTDevice.class类文件，如图8-34所示。打开这个文件后，我们发现设备序列号是可以修改的，于是将前面得到的序列号替换到文件中。重新编译、运行软件，刷新我们手中的Spot Trace的固件，这样Spot Trace就成功地克隆了之前获取信号的设备。刷新手机软件，可以看到两条不同的踪迹，这时假如修改警报信息，原设备用户就会受到虚假信息的骚扰，同时，原设备用户的轨迹信息就这样泄露了。当然还可以在原设备的旁边放置干扰器，或者是在不被发现的情况下破坏掉它，这样，真实的轨迹在世界上消失了，我们可以用克隆到的设备彻底地欺骗使用者。如果使用者只是普通用户，则不会有太大的影响，但是如果使用者为军事设备装甲车、货运飞机、假释犯人或动物，那么这样的破坏行为就存在相当大的危害性。

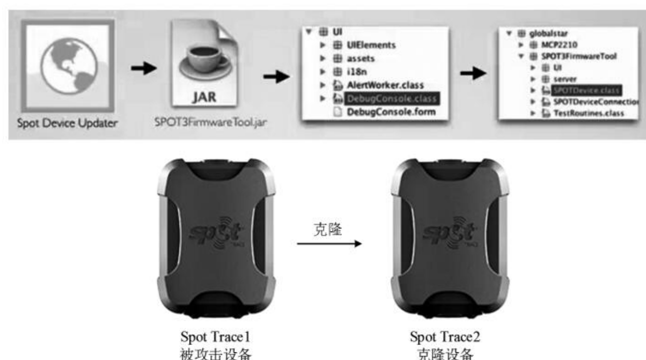


图8-34　克隆Spot Trace设备

2.发射伪造的上行数据

假如我们身边没有这款Spot Trace设备，那么是否可以自行发射信息呢？的确可以，在查到24比特的CRC校验码后，只需对原始信息进行重新编码、扩频，并将数据信息调制到固定载波上就可以发射出去。相对解码器来说更容易，只需一个大功率的发射天线似乎就解决了。而且你可以发射任意的位置信息，然而这样做的后果是你很有可能被FCC请去喝茶。因为当你发射虚假信息时，很有可能会干扰到身边其他重要信息的通信，于是很容易被FCC发现，无执照占用信道发射信息是不合法的。

参考文献

[1] https：//media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEFCON-23-Lin-Huang-Qing-Yang-GPS-Spoofing.pdf

[2] https：//en.wikipedia.org/wiki/Iran%E2%80%93U.S._RQ-170_incident

[3] https：//radionavlab.ae.utexas.edu/

[4] http://www.ted.com/talks/todd_humphreys_how_to_fool_a_gps？language＝zh-cn

[5] http://www.digitaltrends.com/mobile/gps-spoofing/

[6] http://gpsworld.com/drone-hack/

[7] https：//radionavlab.ae.utexas.edu/images/stories/files/papers/summary_financial_sector_implications.pdf

[8] http://military.china.com/important/11132797/20121231/17609033.html

[9] http://www.gps.gov/technical/icwg/IS-GPS-200H.pdf

[10] http://www.wired.com/2011/12/iran-drone-hack-gps/

[11] http://baike.haosou.com/doc/5348881-5584335.html

[12] http://www.gps.gov/technical/icwg/IS-GPS-200H.pdf

[13] 鲁郁.GPS全球定位接收机[M].北京：电子工业出版社，2009

[14] http://www.minicircuits.com/pdfs/ZX60-V82＋.pdf

[15] http://sourceforge.net/projects/gnss-sdr/files/data/

[16]　https：//code.csdn.net/sywcxx/gps-sim-hackrf

[17]　Humphreys T.Statement on the vulnerability of civil unmanned aerial vehicles and other systems to civil GPS spoofing[J].University of Texas at Austin(July18，2012)，2012

[18]　Nielsen J，Broumandan A，Lachapelle G.Method and system for detecting GNSS spoofing signals：U.S.Patent7，952，519[P].2011-5-31

[19]　Scott L.Anti-spoofingamp;authenticated signal architectures for civil navigation systems[C]//Proceedings of the16th International Technical Meeting of the Satellite Division of The Institute of Navigation(ION GPS/GNSS2003).2001：1543-1552

[20]　Wesson K，Rothlisberger M，Humphreys T.Practical cryptographic civil GPS signal authentication[J].Navigation，2012，59(3)：177-193

[21]　Wesson KD，Shepard DP，Bhatti JA，et al.An evaluation of the vestigial signal defense for civil GPS anti-spoofing[C]//Proceedings of the ION GNSS Meeting.2011

[22]　Montgomery PY，Humphreys TE，Ledvina BM.A multi-antenna defense：Receiver-autonomous GPS spoofing detection[J].Inside GNSS，2009，4(2)：40-46

[23]　https：//www.youtube.com/watch？v＝o0-ekeLYz9I

第9章　无线安全研究工具——GNU Radio

在讨论了从近距离的RFID到远距离的卫星通信等各种无线通信系统之后，我们将要在这一章介绍用于无线安全研究的工具，即软件无线电技术。

本章内容大部分取材于GNU Radio官方网站最新的教程。除了GNU Radio，也会简单介绍相关的硬件，以及其他的软件无线电软件。

9.1 软件无线电技术

软件无线电，即Software Defined Radio（SDR），有时也称为Software Radio。

如图9-1所示是典型的软件无线电处理流程图。为了理解无线电的软件模块，首先需要理解与其关联的硬件。在图中的接收路径上，能够看到有天线、RF前端、ADC和代码。ADC是一个连接连续的“Analog”自然世界和离散的“Digital”世界的桥梁。

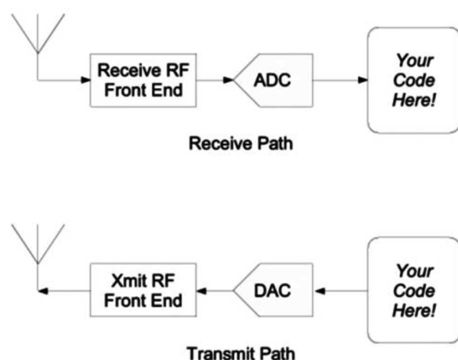


图9-1　典型的软件无线电处理流程图

ADC有两个主要特性：采样率和动态范围。采样率是ADC测量模拟信号的速度；动态范围是ADC区最小信号值和最大信号值的精度，这决定了ADC数字信号输出的比特数（位数）。例如，8位的AD转换器最多能代表256个信号层次，而16位的转换器能够代表65536个层次信号。总的来说，ADC的物理特性和价格决定了采样率和动态范围。

1927年，出生于瑞典的物理和电子学家Harry Nyquist提出了如果想使AD转换没有混叠现象发生，那么采样率至少是目标信号带宽的2倍。这就是著名的奈奎斯特定律。

假设我们要处理一个低通信号，信号带宽是0~f_{max}，按照奈奎斯特理论，采样率必须至少是2*f_{max}。如果ADC的采样率是20MHz，但是我们想收听92.1MHz的FM电台，该怎么办呢？答案是使用RF前

端，接收机的RF前端能够把它接收到的高频段信号下变频到一个低频段信号后输出。例如，如果能让RF前端把90～100MHz频段内的信号下变频到0～10MHz低频段范围内，那么20MHz的ADC就能够派上用场了。

在大多数情况下，我们把RF前端当作一个信号控制的黑盒子，负责处理输入信号的中心频率，使信号在高频和低频之间转换。举一个具体的例子，一个调制解调器的调制模块能够把50～800MHz之间的6MHz带宽的信号下变频到一个中心频率是0Hz的信号输出。这个输出的中心频率通常称作中频（IF）。中频为0的接收机，称为零中频接收机。随着射频芯片和ADC芯片越来越强大，零中频接收机也变得越来越常见。

其实，如果ADC的采样率相对于射频信号的频率差别不是很大的话，RF前端也能够被去掉。有一个GNU Radio用户已经成功地使用一个100英尺（1英尺等于12英寸，合0.305米）的天线直接连接到20MHz采样率的ADC上收听到了AM（300KHz～3MHz）和短波（3～30MHz）广播。

图9-1中的最后一个模块“Your Code Here！”，就是软件代码所在的部分。在广义的软件无线电概念中，软件代码指各种“可编程”的代码，不仅是运行在CPU上的代码，也包括DSP和FPGA平台上的代码。但在本书中，主要指运行在CPU上的代码。因为CPU是最广泛使用的编程平台，运行在CPU上的软件无线电代码最方便移植。

9.1.1 SDR的强大能力

在互联网行业有句话："Software is eating the world"。

这是网景公司的创始人马克·安德森（Marc Andreessen）说的。我想这句话不仅针对互联网公司，对整个工业界都是成立的。无线通信行业，有了SDR的进入，正在发生着快速的变化。

太早之前的故事，这里就不细说了，感兴趣的读者可以读一下维基百科中"Software Defined Radio"这一词条。2000年以后，推动SDR技术的两大功臣，是GNU Radio和USRP这对搭档。

USRP是2004年由Matt Ettus开发的，公司就命名为Ettus Research。在USRP出现之前，SDR都是基于一些价格十几万元人民币的PCI板卡。USRP一下子把SDR的价格拉低到了几千元人民币。后面的章节将详细介绍USRP。

GNU Radio最早出现在2001年，2004年重写了整个项目，配合USRP，实现了一套完整的软硬平台。这对搭档从此书写了SDR的新历史，使SDR成为真正普及的工具。

SDR可以极大地"放大"程序员的能力。

2009年，利用USRP，OpenBTS（http://openbts.org/）这个项目第一次用SDR实现了一个真正能用的GSM基站。而且OpenBTS是开源的，使得每一个程序员只要有USRP，就可以搭建一个GSM小基站。后来，使用SDR来实现基站的尝试就一直没有停止过。2012年，著名的法国程序员Fabrice Bellard（http://www.bellard.org/）以一人之力，开发出了基于USRP＋CPU结构的完整的LTE基站的功能；而在传统的电信设备制造企业中，开发一个基站至少需要多个软硬件团队相互配合。毫不夸张地说，Fabrice Bellard一个人相当于100个普通程序员的生产力。

9.1.2　SDR的用途

由于软件无线电具有能够快速开发、迭代更新的特点，所以它特别适合用在一些定制化的应用中。例如：

- 学生和研究人员用它来研究无线信号处理算法，研究新的通信协议。因为所有的通信协议，从上至下都是PC上的软件代码。你可以像使用普通代码一样快速自如地修改、编译和运行，可以灵活地在多个协议层之间互操作。当你撰写学术论文时，这些真实的实验结果常常能够作为理论分析的强有力支撑。

- 创业型小公司或者学校里做产品开发类项目的人员，通常用它来开发原型设备POC（Prototype of Concept）。比如做一个支持多种制式的家庭网关，因为所有的东西都是“软”的，所以开发起来非常快，出现问题时也容易修改。

- 用来做高校里的教学用实验平台。比如做通信原理实验，现在大部分实验都是用MATLAB仿真来做的，当有了GNU Radio后，你就可以看到真正的信号星座图、频率漂移等现象。而且它可以是一个远程的平台，供很多学生同时使用。

- 业余无线电爱好者，他们用GNU Radio来搭建自己的电台。还有些射电天文学的爱好者，用SDR搭建自己的射电天文站。2014年，有一个民间项目ISEE3Reboot Project，利用USRP和GNU Radio，与一个已经发射到太空60年之久的ISEE3卫星通信，试图重启它的动力系统（http://spacecollege.org/isee3/）。

- 黑客！这是SDR用户中很大的一个群体。安全公司以及政府的无线电监控部门、军方的实验室都对此很有兴趣。目前SDR技术已经广泛地使用在各个频段，用于无线电信号监听、逆向分析、欺骗攻击、监控防御等，这也是本书的主题。

9.2 GNU Radio简介㊟

上一节介绍了SDR。SDR的应用可以有很多种实现方式，GNU Radio是其中一种。

GNU Radio是一套自由开源软件工具，它提供了各种信号处理模块，可用于实现软件无线电系统。它可以外接一些低成本的射频硬件，从而真正实现有无线信号发射接收的完整系统；也可以在没有射频硬件的情况下，运行一些仿真程序。GNU Radio已经被广泛用于科研、商业应用和业余爱好者的活动中。

用手机操作系统来类比，手机可以有很多种操作系统，如iOS、安卓、Windows、Linux；GNU Radio有点像安卓系统，提供了一套运行和开发的平台，屏蔽了底层硬件的细节，可以支持多种射频硬件，而且是完全开源的。

- 使用GNU Radio能做什么？

GNU Radio几乎能完成所有的无线信号处理功能。你可以写一个程序，从接收到的数字信号流中读取数据；也可以把数据推送给发射信号流，然后从射频硬件发射出去。GNU Radio有滤波器、信道编码、同步模块、均衡器、解调器、信道解码，还有许多其他的模块。在GNU Radio的术语中，称之为“block”，在这里译为“模块”吧。更重要的是，GNU Radio提供了一种方法，把这些模块连接在一起，而且能够自动地管理信号从一个模块流动到另一个模块。扩展GNU Radio也非常容易，如果发现缺少某种模块，则可以自己编写一个新模块，把它添加进去。

既然GNU Radio是软件，那么它只能处理数字信号。一般来说，接收和发射的信号，都用复数基带信号这种数据类型来表示。外部的射频硬件完成数字信号与模拟信号之间的转换，还负责完成信号在基频和射频之间搬移。在GNU Radio内部的模块之间，数据可以以各种格式流动，比如比特、字节、向量、数据包（burst），或者更复杂的一些数据格式。

GNU Radio是用Python语言写的，其中对计算性能要求很高的模块，是用C++语言写的。因此，GNU Radio既满足了易用性，又满足了无线系统对实时性和高信号吞吐量的要求。

○ 必须懂得编程才能使用GNU Radio吗？

当然最好是懂得编程。然而，GNU Radio也提供了其他的方法，方便不懂得编程的人使用。首先，GNU Radio有一个叫GNU Radio Companion（GRC）的工具，具有完全图形化的界面，类似于Simulink，只需要拖放模块，就可以搭建一个无线系统；其次，GNU Radio还提供了一些现成的应用程序，这些程序可以实现一些基本的操作，例如录制一段射频信号、做频谱分析等。

如果你想扩展GNU Radio的功能（例如增加新的功能模块），那么就必须懂得编写代码。这比使用GRC难多了，简单点的模块使用Python编程就足够了；对计算性能要求高的模块，就得用到C++编程。

○ GNU Radio使用哪种开源许可？

GNU Radio采用GPL3.0（General Public License version3.0），软件的版权属于自由软件基金会。

关于几种开源软件许可的差别，在此引用阮一峰博文中的一张图（见图9-2），能够简明、清晰地说明GPL许可的特点。

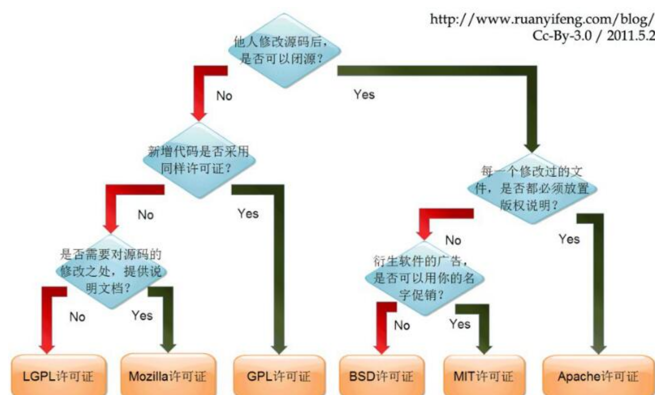


图9-2　几种开源软件许可的差别

由此可知，GNU Radio修改源码之后，不能闭源；新增的代码也必须使用同样的GPL许可证。

○ 人们用GNU Radio做了些什么？

如果你安装了GNU Radio，就会看到其中包含了很多实用的范

例，比如有发射数字信号的例子，有接收模拟信号波形的例子，还有很多其他的例子。除了GNU Radio自己收录的这些范例程序外，还有CGRAN（http://www.cgran.org/）网站，收录了大量的GNU Radio项目。另外，http://gnuradio.org/redmine/projects/gnuradio/wiki/OtherCode也收录了一些项目。

- 数字信号处理、基带、同步等都是什么？

如果你问出这个问题，则说明缺乏无线通信的基础知识，那么对你来说使用GNU Radio就很困难了。因为它毕竟只是一个工具，你必须懂得无线通信和信号处理的原理，才能够用工具搭建一个系统。不过也不必被吓倒，找一些相关书籍学习一下，再配合GNU Radio的代码，理论结合实践，相信你一定可以进步神速。

1. 这部分内容整理自http://gnuradio.org/redmine/projects/gnuradio/wiki/WhatIsGR。返回

9.3 GNU Radio支持的硬件工具

虽然GNU Radio在没有硬件支持的情况下，也可以做一些仿真，不过那样就太无趣了。所以每一个想玩GNU Radio的小伙伴，至少装备一种硬件吧。这一节我们就简单介绍几种常用的GNU Radio能够支持的硬件工具。

9.3.1 USRP

第一种要介绍的硬件当然是大名鼎鼎的USRP。USRP由Ettus Research（www.ettus.com）公司制造。这个公司是2004年Matt Ettus从GNU Radio项目中独立出来之后创立的。Matt Ettus和USRP1如图9-3所示。

USRP是开源硬件，原理图、固件代码、上位机代码都是开源的。2010年，Ettus Research公司被NI（National Instruments）收购，但仍然保持独立的品牌，并保持开源。

USRP经过十几年的发展，已经有多个产品系列，根据接口类型不同，有：

- USRP X系列—X系列使用10G以太网接口，是USRP中性能最强的系列，可以支持到120MHz射频带宽。



图9-3　Matt Ettus和USRP1[注]

- USRP N系列—N系列使用1G以太网接口，N系列在X系列出现以前，是非常热门的一款产品，用户量很大。

- USRP B系列—B系列使用USB接口，早期是USB2.0接口。第一代USRP产品，叫USRP1，就是使用USB2.0接口的。现在都在使用USB3.0接口。USB接口的优点是，接口数量多，而且带有USB接口的

设备非常多，除了计算机以外，手机、小型嵌入式设备都有USB接口。所以USB接口是一种广泛存在、非常方便的接口。

- USRP E系列—E系列是独立运行的USRP设备，自带ARM处理器，无须上位机。这个系列特别适合那些需要独立运行的小系统。

今天，USRP正在朝两个方向同时发展，其中一个方向是追求高速高性能，代表就是X系列，如图9-4所示。



图9-4　USRP X310

一个X310的高度是1U，两个X310正好可以塞进一个1U的机架。每个X310可以配上两个射频子板。X310不仅可以支持高射频带宽，而且还带有强大的FPGA，可以分担一部分计算任务。

Ettus Research公司开发了一种软件工具，叫作RFNoC（RF Network on Chip），类似于GRC那样的图形界面，方便开发FPGA程序。图9-5显示的是一个频谱分析程序，这个程序显示出一种具有漂亮的磷光效果的叠加FFT频谱（见图9-6），基带信号的采样率高达100MS/s。如果只用CPU来计算的话，运算速度是来不及的，因此这里用的是FPGA的计算能力。把RFNoC的几个模块拖到一起，就可以很容易地搭建出一个这样的程序。

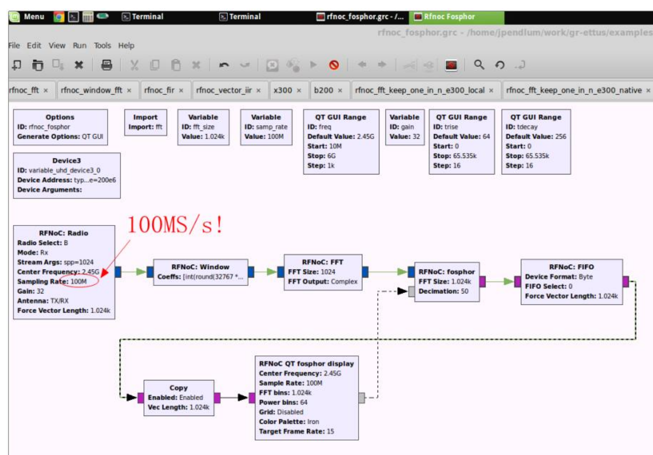


图9-5　频谱分析程序

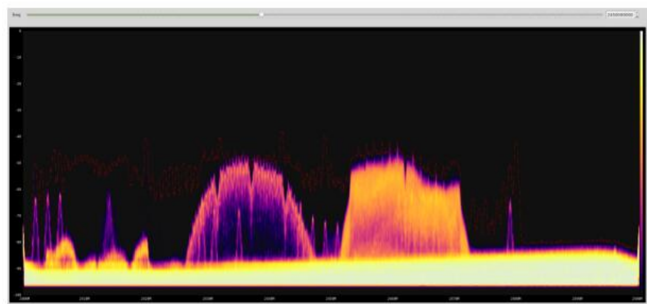


图9-6　具有漂亮的磷光效果的叠加FFT频谱

可见，RFNoC代表了USRP正在追求高速、高计算能力这个方向。

另一个方向是“小型化”。Ettus Research在2015年9月发布了一款新的B系列USRP设备，是B200/210的迷你版，尺寸如同名片大小，如图9-7所示。

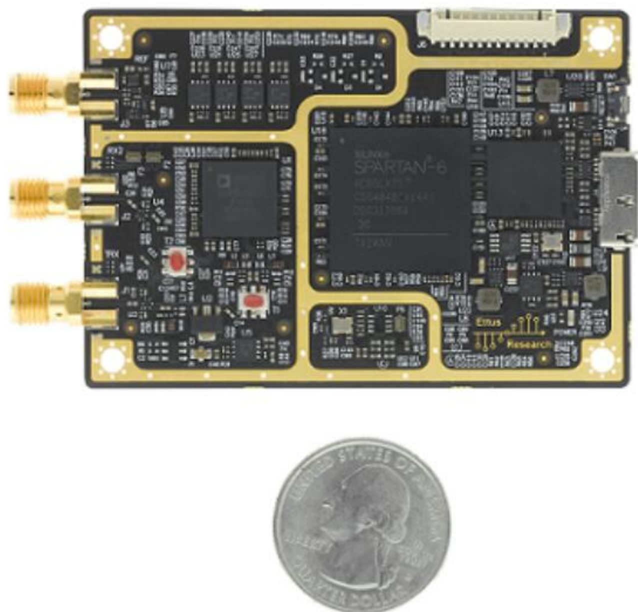


图9-7　B系列USRP设备

B200mini与B200的功能完全相同，它的小身材使其非常便携，也很容易与主机设备集成在一起。GNU Radio正在研发Android版本，B200mini加一部安卓手机，配合GNU Radio安卓版本，就是一个最小的SDR系统（见图9-8）。预计黑客们会非常喜欢这个迷你身材的小USRP。

USRP的射频指标与产品型号有关，常用的几款USRP的射频载波都支持从几十MHz到几GHz，射频带宽8~120MHz，发射功率10~100dBm，在此就不详细介绍了。

USRP的驱动程序称为UHD（USRP Hardware Driver），它是一套独立的驱动程序，与GNU Radio无关。有很多应用，例如OpenBTS和许多LTE软基站项目，都是直接调用UHD接口的，不依赖于GNU Radio框架。

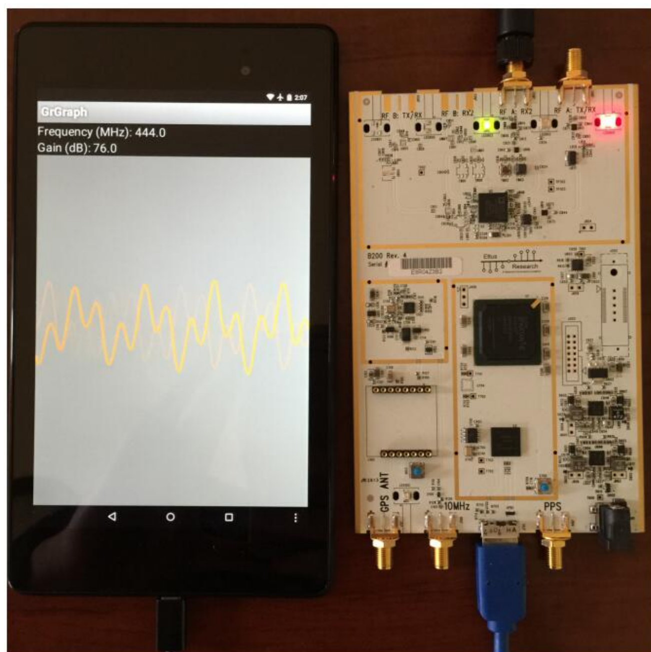


图9-8　Nexus7和一个USRP B200（不是mini版），运行GNU Radio安卓版本

除了GNU Radio以外，USRP还支持NI自家的LabVIEW，也支持MATLAB。不过，因为不是免费开源的，用的人少，USRP社区里讨论后两者的人非常少。

1. 图片出处：http://www.theamphour.com/the-amp-hour-101-quality-quadrature-quidam/。返回

9.3.2　RTL-SDR

除了USRP，另一种人气颇高的硬件要数RTL-SDR（http://www.rtl-sdr.com/）了，如图9-9所示。

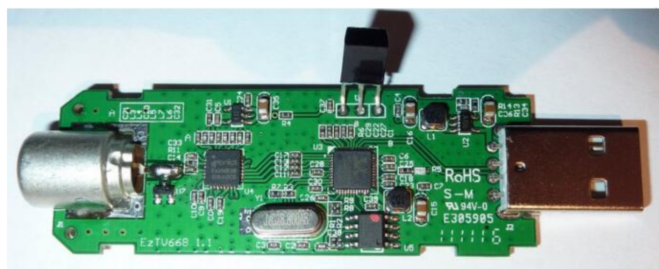


图9-9　RTL-SDR

RTL-SDR是一种特别便宜的软件无线电硬件。它原本是一个基于RTL2832U芯片的DVB-T Dongle，用来收看电视，俗称“电视棒”。后来发现这种芯片可以直接读取原始的I/Q数据，于是只要安装一个新的驱动程序，电视棒就可以被用来作为宽带的软件无线电接收机。

这样一个电视棒的价格大约20美元，在淘宝上甚至可以找到30~40元人民币一个的价格。用它配合电脑，就可以实现一个频谱扫描器的功能。过去，这类扫描器需要几百甚至几千美元才能买到。

（1）RTL-SDR的射频频率范围

这取决于使用了哪一种调谐器（Tuner），也就是变频器（见表9-1）。

表9-1 调谐器及对应的频率范围

调谐器	频率范围
Elonics E4000	52～2200MHz，1100～1250MHz 空出来
Rafael Micro R820T	24～1766MHz
Rafael Micro R828D	24～1766MHz
Fitipower FC0013	22～1100MHz
Fitipower FC0012	22～948.6MHz
FCI FC2580	146～308MHz 和 438～924MHz

可见，频率范围最宽的是Elonics E4000。

（2）RTL-SDR的采样率和ADC精度

RTL2832U的ADC是8比特I/Q两路采样。理论上，最高的采样率是3.2MS/s。然而，如果工作在这个采样率下，RTL-SDR会不太稳定，会丢失数据。因此，目前能达到的、无数据丢失的最高采样率是2.56MS/s。

除了GNU Radio，RTL-SDR还支持很多其他的软件。大部分软件包都是以“librtlsdr”这个库为基础的。源代码位于http://cgit.osmocom.org/rtl-sdr/，

这套代码不仅包含库函数，还有一些命令行工具，例如rtl_test、rtl_sdr、rtl_tcp、rtl_fm。这些命令行工具可以用来测试RTL-SDR设备连接是否正常，还可以完成基本的信号采集功能。因为RTL-SDR使用USB接口与主机连接，所以librtlsdr库依赖于libusb库。

在前面的章节中，有很多地方用到了RTL-SDR做信号分析，比如车钥匙信号分析、停车杆控制信号分析、胎压传感器信号分析。我们很喜欢用它配合HDSDR（http://www.hdsdr.de/）这个软件，做基础的频谱分析和简单信号解调的工具。除以上这些用法之外，RTL-SDR还可以用来做很多事情，下面来看一下这个长长的列表：

Listening to unencrypted Police/Ambulance/Fire/EMS conversations.

Listening to aircraft traffic control conversations.

Tracking aircraft positions like aradar with ADSB decoding.

Decoding aircraft ACARS short messages.

Scanning trunking radio conversations.

Decoding unencrypted digital voice transmissions.

Tracking maritime boat positions like aradar with AIS decoding.

Decoding POCSAG/FLEX pager traffic.

Scanning for cordless phones and baby monitors.

Tracking and receiving meteorological agency launched weather balloon data.

Tracking your own self launched high altitude balloon for payload recovery.

Receiving wireless temperature sensors and wireless power meter sensors.

Listening to VHF amateur radio.

Decoding ham radio APRS packets.

Watching analogue broadcast TV.

Sniffing GSM signals.

Using rtl-sdr on your Android device as aportable radio scanner.

Receiving GPS signals and decoding them.

Using rtl-sdr as aspectrum analyzer.

Receiving NOAA weather satellite images.

Listening to satellites and the ISS.

Radio astronomy.

Monitoring meteor scatter.

Listening to FM radio , and decoding RDS information.

Listening to DAB broadcast radio.

Use rtl-sdr as apanadapter for your traditional hardware radio.

Decoding taxi mobile data terminal signals.

Use rtl-sdr as ahigh quality entropy source for random number

generation.

Use rtl-sdr as anoise figure indicator.

Reverse engineering unknown protocols.

Triangulating the source of asignal.

Searching for RF noise sources.

Characterizing RF filters and measuring antenna SWR.

从其中的关键词就可以看出，有很多种系统，从对讲机、广播电台，到移动通信、卫星通信，有非常丰富的应用场景。原因是，RTL-SDR这种设备所覆盖的频段，是最好的无线电频段，波长短（天线短，容易发射）、传播损耗低（传播距离远），所以这段宝贵的频谱，被精打细算地分配给了各种系统。于是，我们就可以用电视棒在各种各样的系统中玩耍了。

总而言之，对一个软件无线电初学者来说，RTL-SDR作为入门的硬件设备，是一个非常好的选择。

9.3.3 HackRF

HackRF是一种最便宜、收发功能齐全的SDR硬件，如图9-10所示。



图9-10 HackRF的外观

说它是最便宜的，因为它只需要大约300美元；但它比RTL-SDR贵，因为RTL-SDR只能接收，不能发射，而HackRF既能接收又能发射。所以，如果你想尝试一些既要接收又要发射的实验，那么成本最低的选择就是HackRF了。

2013年上半年，硬件黑客Michael Ossmann和他的小伙伴Jared Boone一起开发了HackRF这个低成本的SDR硬件。在2013年BlackHat和DEFCON大会上，他们在演讲中向大家介绍了这种硬件，而且同时在Kickstarter上发起了众筹，打算量产HackRF，只花了6个小时，钱就筹够了！这种硬件就这么火起来了。

HackRF也是开源的，跟USRP类似，国内也有生产HackRF的团队。不同的是，HackRF是完全开源的软件无线电射频前端，从原理图到PCB图、从驱动程序到单片机固件，甚至连加工制板所需要的工艺要求，全部以GPL协议无保留地发布，这对于我们学习研究软件无线电无疑是一件非常值得庆幸的事情。

（1）HackRF支持的射频频段

HackRF（https：//greatscottgadgets.com/hackrf/）支持

1MHz~6GHz这么宽的频段。因为HackRF的开放性，它有一些其他版本，支持不同的频率范围。例如HackRF Blue（http://hackrfblue.com/），号称可以支持200KHz~7.2GHz（没有额外的上下变频器）。

（2）HackRF的采样率和ADC/DAC精度

HackRF的采样率最高可以达20MS/s，8比特量化（8比特I路加上8比特Q路）。

（3）主机接口类型

HackRF使用USB2.0接口。

9.3.4　bladeRF

HackRF虽然既能收也能发，但却是半双工的，也就是说，收发不能同时进行。bladeRF是一种全双工的SDR硬件（https：//www.nuand.com），如图9-11所示。



图9-11　bladeRF

bladeRF几乎是与HackRF同时出现的。2012年下半年，Robert Ghilduta带领Nuand团队开发了bladeRF的原型；2013年1月，在Kickstarter上发起了众筹，很快在2月底就筹足了资金。bladeRF也在BlackHat和DEFCON会议上做了推广和销售，所以它也是一开始就面向硬件黑客群体的。

（1）bladeRF支持的射频频段

bladeRF可以支持300MHz~3.8GHz。

（2）ADC/DAC采样率和精度

采样率最高40MS/s。12比特量化。

（3）主机接口类型

bladeRF使用USB3.0接口。

从以上三项指标来看，除了射频范围，bladeRF的指标都超过了

HackRF。bladeRF选择了USB3.0接口，可以支持更大的基带带宽。最重要的是，bladeRF是全双工的，因此可以支持OpenBTS这样的应用。

bladeRF对多天线的支持也相当不错，通过同步时钟电缆连接多个bladeRF，可以支持2天线和4天线MIMO。bladeRF有两种型号：x40有一个容量略小的FPGA，40KLE；x115有一个容量较大的FPGA，115KLE Cyclone IV。与USRP类似，FPGA用来处理一些高运算量的任务。bladeRF上还配备了一个ARM处理器，使它可以脱离主机单独运行。

在软件方面，bladeRF除了支持Linux系统，还支持Windows和Mac，特别是它可以支持MATLAB。

总的来说，bladeRF更接近于USRP，像USRP的低成本版本。如果说USRP比较学术范，HackRF黑客范，那么bladeRF则是介于两者之间的类型。

这一节我们讨论了4种常用的SDR硬件。下面的表9-2总结了这4款硬件的主要指标和价格，供读者参考。

表9-2 4种常用的SDR硬件的主要指标和价格

	RTL-SDR	HackRF One	bladeRF x40	USRP B200mini
频段	52~2200MHz	1MHz~6GHz	300MHz~3.8GHz	70MHz~6GHz
带宽	2.56MS/s	20MS/s	40MS/s	56MS/s
双工类型	只能接收	半双工	全双工	全双工
位宽	8bit	8bit	12bit	12bit
主机接口	USB 2.0	USB 2.0	USB 3.0	USB 3.0
FPGA	N/A	CPLD.No FPGA	40KLE AlteraCyclone4	Xilinx Spartan-6XC6SLX75
微控制器	RTL2832U	LPC4320	CypressFX3	CypressFX3
开源程度	无官方公开版本，有逆向版本[3]	完全开源	原理图、HDL 代码、上位机代码开源	原理图、HDL 代码、上位机代码开源
价格	$20	$300	$420	$675

3 GGToshi提供的一个逆向后版本，http://ggtoshi.at.webry.info/201406/article_6.html。

9.4 GNU Radio安装

现在进入GNU Radio正题。准备好了硬件，就可以开始安装软件了。

GNU Radio有很多种安装方法，在官方网站的安装指南页面（http://gnuradio.org/redmine/projects/gnuradio/wiki/InstallingGR），可以看到这些内容：

- Linux安装
- Windows安装
- Mac OS X安装
- 使用Live DVD镜像
- 从源码安装

这里我们只关注Linux系统下的安装，因为Linux下的用户最多，网络资源也最丰富。下面以Ubuntu系统为例。

不推荐使用apt-get install的方式自动安装GNU Radio，因为通常Ubuntu默认安装的版本总是非常老旧的。一旦想换成新版本，旧版本又很难卸载干净，造成版本冲突。所以我们还是从源码安装吧。

9.4.1 从源码手动安装

从源码安装，是安装GNU Radio最靠谱的一种方法。因为各种自动安装脚本很可能因为更新或维护不及时，产生各种错误。而如果从源码安装，就只依赖于两个因素：Ubuntu系统的源码和GNU Radio的源码。

下面就以GNU Radio＋USRP这对搭档为例，来介绍从源码手动安装GNU Radio的步骤。

1.安装UHD

因为硬件使用的是USRP，所以我们首先要安装UHD。推荐从Ettus Research官网（https：//github.com/EttusResearch/UHD/tags）下载UHD最新的release版本。

例如，我们下载了UHD3.9.2的压缩包，把它解压到某个目录下。

http://files.ettus.com/manual/page_build_guide.html这个地址是UHD的安装指南，根据这里的说明，需要为UHD先安装一些依赖库。

```
bui sudo apt-get install libboost-all-dev libusb-1.0-0-dev python-mako doxygen python-dc
```

安装好之后，就可以产生UHD的makefile了。执行以下命令：

```
cd ＜uhd-repo-path＞/host
mkdir build
cd build
cmake ../
```

注意检查cmake之后的结果，看看是不是所需要的模块都被enabled了。比如cmake之后输出的最后一部分是：

```
-- ##################################################
```

```
-- # UHD enabled components
-- #######################################################
--   * LibUHD
--   * LibUHD - C API
--   * Examples
--   * Utils
--   * Tests
--   * Manual
--   * API/Doxygen
--   * Man Pages
--   * USB
--   * USRP1
--   * USRP2
--   * B100
--   * X300
--   * B200
--   * OctoClock
--
-- #######################################################
-- # UHD disabled components
-- #######################################################
--   * GPSD
--   * E100
--   * E300
```

可以看到disabled部分有GPSD、E100、E300。在这里我们不会用到E系列的硬件，因此不必介意这些模块被禁用了。

cmake没有错误之后，就可以开始编译安装了。

```
make
make test
sudo make install
sudo ldconfig
```

安装完成之后，需要下载USRP的固件。因为每个UHD版本都有自己相应的固件，因此建议在安装完UHD之后，使用UHD自己的脚本下载对应的固件。固件文件较大，下载会比较耗费时间。

```
cd /usr/local/lib/uhd/utils/
sudo ./uhd_images_downloader.py
Images destination:      /usr/local/share/uhd/images
Downloading images from: http://files.ettus.com/binaries/images/uhd-images_003.009.0
Downloading images to:   /tmp/tmpgQCxRO/uhd-images_003.009.002-release.zip
26296 kB / 26296 kB (100%)

Images successfully installed to: /usr/local/share/uhd/images
```

可以看到uhd_images_downloader.py程序运行完之后，将把固件放在/usr/local/share/uhd/images目录下。

至此，UHD的安装就算完成了。此时如果你给电脑插上USRP，使用uhd_usrp_probe.py命令，就可以看到连接的USRP信息。

2.安装GNU Radio

接下来就可以安装GNU Radio了。首先下载最新的Release版本，代码位于http://gnuradio.org/releases/gnuradio/，然后把压缩包解压到某个文件夹下。

按照http://gnuradio.org/redmine/projects/gnuradio/wiki/UbuntuInstall这个地址中的指导，安装GNU Radio的依赖库。

对于Ubuntu14.04系统来说，就是执行以下命令安装依赖库。

```
sudo apt-get -y install git-core cmake g++ python-dev swig \
pkg-config libfftw3-dev libboost1.55-all-dev libcppunit-dev libgsl0-dev \
libusb-dev libsdl1.2-dev python-wxgtk2.8 python-numpy \
python-cheetah python-lxml doxygen libxi-dev python-sip \
libqt4-opengl-dev libqwt-dev libfontconfig1-dev libxrender-dev \
python-sip python-sip-dev
```

这些依赖库能否正确安装，决定了接下来的cmake能不能正确完成。

```
$ mkdir build
$ cd build
$ cmake ../
```

注意查看cmake的结果，是不是所需要的模块都被enabled了。比如cmake的结果是这样的：

```
-- ################################################################
-- # Gnuradio enabled components
-- ################################################################
--   * python-support
--   * testing-support
--   * volk
--   * doxygen
```

```
--   * gnuradio-runtime
--   * gr-ctrlport
--   * gr-blocks
--   * gnuradio-companion
--   * gr-fec
--   * gr-fft
--   * gr-filter
--   * gr-analog
--   * gr-digital
--   * gr-dtv
--   * gr-atsc
--   * gr-audio
--   * * alsa
--   * * oss
--   * gr-channels
--   * gr-noaa
--   * gr-pager
--   * gr-qtgui
--   * gr-trellis
--   * gr-uhd
--   * gr-utils
--   * gr-video-sdl
--   * gr-vocoder
--   * gr-fcd
--   * gr-wavelet
--   * gr-wxgui
--
-- ##################################################
-- # Gnuradio disabled components
-- ##################################################
--   * sphinx
--   * gr-comedi
--   * gr-zeromq
```

只有我们不关心的三个模块被disabled了。

OK，接下来就可以编译安装了：

```
$ make && make test
$ sudo make install
```

默认的安装路径是/usr/local目录。

9.4.2　使用PyBOMBS安装GNU Radio

如果你觉得从源码手动安装太麻烦，在源码安装指南中，GNU Radio提供了PyBOMBS工具。这个工具把所有的GNU Radio相关的软件都收集到一起，方便用户安装。可以尝试这种安装方法。

PyBOMBS（Python Build Overlay Managed Bundle System）是一套GNU Radio的安装管理系统，用来解决各个软件库之间的依赖问题。收集out-of-tree projects，也是PyBOMBS的一个主要目的。

使用PyBOMBS安装，最简单的步骤是这样的：

```
git clone --recursive git://github.com/pybombs/pybombs
cd pybombs
./pybombs install gnuradio
```

执行第三条命令以后，会出现很多提示问题，一路回车，都选择默认选项就可以了。之后你就可以干别的去了（偶尔需要输入一下root密码），PyBOMBS会花很长时间下载，然后安装这些软件。

首先PyBOMBS会安装一些依赖的系统库，例如python、fftw、qt等。接下来它会下载gnuradio、uhd、gr-osmosdr等软件的源码，然后编译安装。这些源码会放在src目录下。

在默认情况下，PyBOMBS把所有的软件都安装在~/target目录下。安装完之后，产生一个“环境”文件：

```
./pybombs env
```

然后运行设置环境参数的脚本：

```
source $prefix/setup_env.sh
```

现在整个安装过程就完成了。我们可以运行GNU Radio程序了。如果运行：

```
./app_store.py
```

可以查看到已经安装了哪些软件模块，如图9-12所示。

图9-12　已经安装的软件模块

可以看到gnuradio肯定是已经安装了的。默认还安装了uhd驱动，gr-osmosdr也安装了，其中包含rtl-sdr、bladeRF和HackRF三种硬件的驱动。因此，前面提及的4种硬件都可以完美支持了。

现在PyBOMBS中已经有非常多的项目了，如果你有自己的项目想加入PyBOMBS，也可以提交申请。

9.4.3 如何更新软件版本

GNU Radio的版本经常有更新，所以很可能过几个月就需要更新一次版本。此时比较安全的做法是，先卸载GNU Radio和它依赖的其他软件包（uhd和gr-osmosdr），升级GNU Radio源码，然后编译安装所有的这些软件包。注意：一定要先make uninstall，把GNU Radio先卸载干净，否则很容易引起版本冲突。

9.5　安装好之后可以做的第一件事

如果使用的是源码安装，bin目录位于/usr/local下；如果使用的是PyBOMBS的默认路径安装，在target下的bin目录中有一些可执行程序。

9.5.1 如果有硬件

如果你手头有硬件，比如USRP，那么立刻就可以尝试一些简单的程序。插上USRP之后，运行：

```
sudo uhd_usrp_probe
```

输出应该是当前连接的USRP的参数：

```
test@ub1404:/usr/local/bin$ sudo uhd_usrp_probe
linux; GNU C++ version 4.8.2; Boost_105400; UHD_003.008.000-18-g864f84b5

-- Operating over USB 3.
-- Initialize CODEC control...
-- Initialize Radio control...
-- Performing register loopback test... pass
-- Performing register loopback test... pass
-- Performing CODEC loopback test... pass
-- Performing CODEC loopback test... pass
-- Asking for clock rate 32.000000 MHz
-- Actually got clock rate 32.000000 MHz
-- Performing timer loopback test... pass
-- Performing timer loopback test... pass
  _____________________________________________________
 /
|       Device: B-Series Device
|     _____________________________________________________
|    /
|   |       Mboard: B210
|   |   revision: 4
|   |   product: 2
|   |   serial: F61903
|   |   FW Version: 7.0
|   |   FPGA Version: 4.0
|   |
|   |   Time sources: none, internal, external, gpsdo
|   |   Clock sources: internal, external, gpsdo
|   |   Sensors: ref_locked
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 0
|   |   |   Freq range: -16.000 to 16.000 MHz
|   |     _____________________________________________________
```

```
|   |   /
|   |   |       RX DSP: 1
|   |   |   Freq range: -16.000 to 16.000 MHz
|   |   _____________________________________________________
|   |   /
|   |   |       RX Dboard: A
|   |   |   _____________________________________________________
|   |   |   /
|   |   |   |       RX Frontend: A
|   |   |   |   Name: FE-RX2
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors:
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 73.0 step 1.0 dB
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |   _____________________________________________________
|   |   |   /
|   |   |   |       RX Frontend: B
|   |   |   |   Name: FE-RX1
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors:
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 73.0 step 1.0 dB
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |   _____________________________________________________
|   |   |   /
|   |   |   |       RX Codec: A
|   |   |   |   Name: B210 RX dual ADC
|   |   |   |   Gain Elements: None
|   |   _____________________________________________________
|   |   /
|   |   |       TX DSP: 0
|   |   |   Freq range: -16.000 to 16.000 MHz
|   |   _____________________________________________________
|   |   /
|   |   |       TX DSP: 1
|   |   |   Freq range: -16.000 to 16.000 MHz
|   |   _____________________________________________________
|   |   /
|   |   |       TX Dboard: A
|   |   |   _____________________________________________________
|   |   |   /
|   |   |   |       TX Frontend: A
|   |   |   |   Name: FE-TX2
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors:
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
```

```
| | | |  Connection Type: IQ
| | | |  Uses LO offset: No
| | |   ____________________________________
| | |  /
| | | |     TX Frontend: B
| | | |  Name: FE-TX1
| | | |  Antennas: TX/RX
| | | |  Sensors:
| | | |  Freq range: 50.000 to 6000.000 MHz
| | | |  Gain range PGA: 0.0 to 89.8 step 0.2 dB
| | | |  Connection Type: IQ
| | | |  Uses LO offset: No
| | |   ____________________________________
| | |  /
| | | |     TX Codec: A
| | | |  Name: B210 TX dual DAC
| | | |  Gain Elements: None
```

这是USRP B210。

然后可以运行uhd_fft程序，它是用来做频谱分析的。先看一下它的参数：

```
test@ub1404:/usr/local/bin$ uhd_fft --help
linux; GNU C++ version 4.8.2; Boost_105400; UHD_003.008.000-18-g864f84b5

Usage: uhd_fft [options]

Options:
  -h, --help            show this help message and exit
  -a ARGS, --args=ARGS  UHD device address args , [default=]
  --spec=SPEC           Subdevice of UHD device where appropriate
  -A ANTENNA, --antenna=ANTENNA
                        select Rx Antenna where appropriate
  -s SAMP_RATE, --samp-rate=SAMP_RATE
                        set sample rate (bandwidth) [default=1000000.0]
  -f FREQ, --freq=FREQ  set frequency to FREQ
  -g GAIN, --gain=GAIN  set gain in dB (default is midpoint)
  -W, --waterfall       Enable waterfall display
  -S, --oscilloscope    Enable oscilloscope display
  --avg-alpha=AVG_ALPHA
                        Set fftsink averaging factor, default=[0.1]
  --averaging           Enable fftsink averaging, default=[False]
  --ref-scale=REF_SCALE
                        Set dBFS=0dB input value, default=[1.0]
  --fft-size=FFT_SIZE   Set number of FFT bins [default=1024]
  --fft-rate=FFT_RATE   Set FFT update rate, [default=30]
  --wire-format=WIRE_FORMAT
```

```
                  Set wire format from USRP [default=sc16]
 --stream-args=STREAM_ARGS
                  Set additional stream args [default=]
 --show-async-msg      Show asynchronous message notifications from UHD
                  [default=False]
```

使用uhd_fft来观察GSM频段的信号频谱：

```
sudo uhd_fft –f 940e6 –s 10e6
```

可以看到一个图形化界面，如图9-13所示。

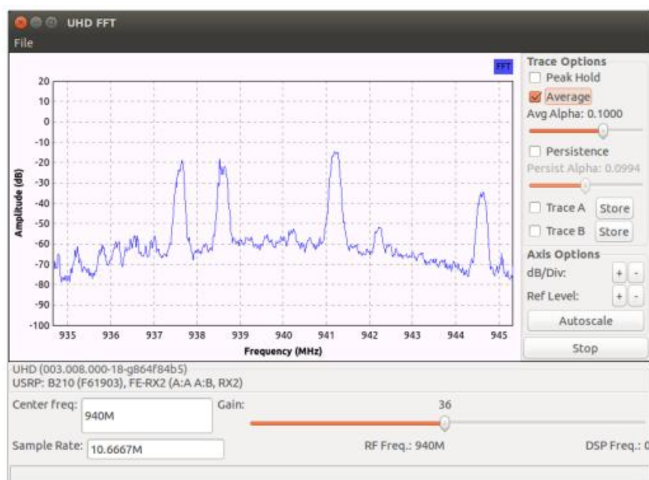


图9-13　使用uhd_fft观察GSM频段的信号频谱

其中突出的信号频谱，就是200KHz宽度的GSM载波。看到这样的界面，就说明USRP、天线都连接正确，软件运行正常。

uhd_fft是一个非常有用的程序，可以帮助我们确认硬件和软件是否正常工作，可以观察频谱当前的状况。在使用GNU Radio的过程中uhd_fft常常被用到。

9.5.2 如果没有硬件

如果你手头没有硬件，那怎么办呢？你可以先体验一下GRC。GRC是一个类似于Simulink的图形化工具，可以设计信号处理流图。如果你对FIR滤波器、数字调制和其他数字信号处理的概念非常熟悉，那么GRC很快就可以上手。

在命令行下运行：

```
gnuradio-companion
```

就会弹出一个图形化窗口。我们先看一个已有的例子。在源码目录gnuradio/grc/examples/simple中，有一个很简单的例子variable_config.grc，如图9-14所示。

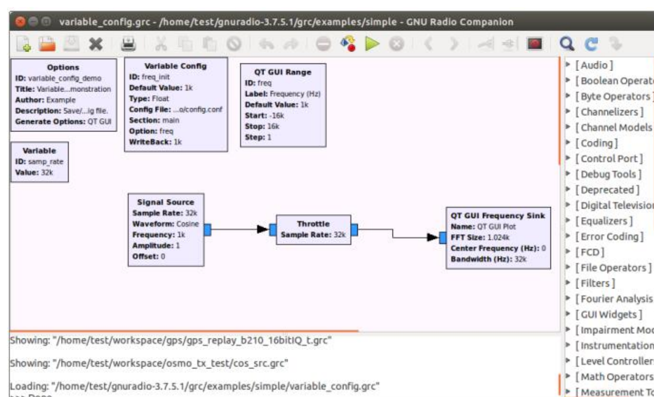


图9-14　variable_config.grc例子

在这个窗口的右侧有各种模块供选择，你可以把所需要的模块从右侧拖到主窗口中。现在主窗口中已经有一个简单的流图，带箭头的线连接三个模块。其中Signal Source是一个信号源，从配置的参数可以看出，它要发出1KHz的余弦波形。Throttle是一个节流阀，用来控制整个流图的速度。Sample Rate设为32K的含义是，经过它的信号，要以32KS/s的速度流过去。Throttle在仿真类型的流图中是非常关键的，它可以控制整个流图模拟真实的采样率工作。如果缺少了它，仿真类型的流图会以CPU的最高速度运行整个流图，很可能会造成仿真

的混乱。最后的QT GUI Frequency Sink是一个频谱分析模块，从配置的参数可见，它将使用1024点FFT，展示32KHz带宽之内的频谱。

主窗口中左上方的四个模块，是一些变量的初始化模块，用来定义这个流图的名字和几个变量的名字。

点击上方工具栏中的绿色三角按钮，就可以运行整个流图。然后就可以看到弹出另一个窗口，显示的是余弦波形的频谱，是一个在1KHz位置的冲击函数，如图9-15所示。

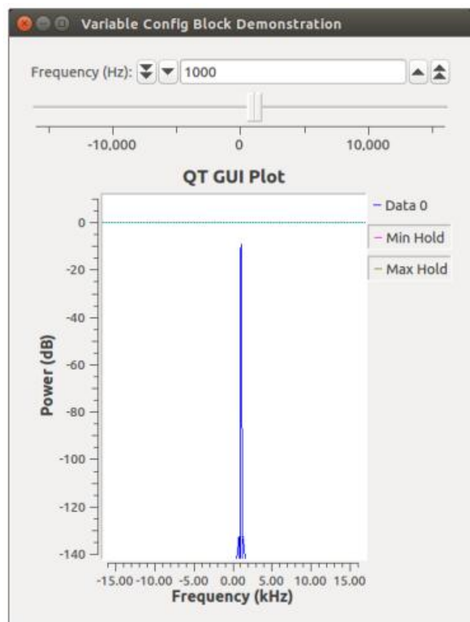


图9-15　余弦波形的频谱

9.6 GNU Radio的一些基本概念注

本节将介绍GNU Radio的最基础概念。

1. 这部分内容来自http://gnuradio.org/redmine/projects/gnuradio/wiki/TutorialsCoreConcepts。返回

9.6.1 流图（flow graph）

流图，顾名思义，就是描述信号如何流动的一张图。大部分GNU Radio程序都是以流图为基础的。流图里的节点，称为“模块”，模块被“线”连接在一起，数据沿着线在模块之间流动。

真正的数字信号处理都是在模块中完成的。每个模块完成一个任务，这样GNU Radio就可以具有灵活性和模块化。模块一般是用C++编写的（有些模块是用Python编写的），我们可以自己编写代码，造出自己的模块。

先来看一个例子。图9-16中有三个模块，数据从左向右流动，首先从一个音频模块中产生出来，然后经过一个低通滤波器，最后写到一个文件里。

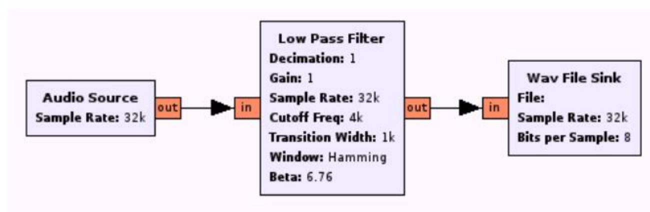


图9-16　音频处理流图

这些模块的端口由带方向的线连接在一起。第一个模块没有输入端口，它只产生信号，所以只有输出端口。这种只有输出端口的模块，叫“source”，中文可以称作“信源”。相反，只有输入而没有输出的模块，叫“sink”，中文可以称作“信宿”（编者：“信宿”这种译法不够好，下文中将保持英文名称sink）。

这个流图的功能是，音频source模块连接到声卡驱动，从声卡驱动获得音频信号。这些信号接着被低通滤波器模块处理，最后被Wav File Sink模块写到一个WAV文件中。

9.6.2　信号流中的颗粒（item）

模块的输入输出的单位，称为item。在上面的例子中，一个item就是声卡驱动输出的一个浮点型数值，也就是一个4字节的float型变量。

item可以是各种类型的变量。最常见的类型有实数信号（一个浮点数）、复数信号（I/Q两路信号，每路一个浮点数）、整型数值、向量。

为了更好地理解这个概念，以FFT为例，对一路信号做FFT分析，然后再存到文件中。进行FFT运算就必须采集一定数量的信号，比如在图9-17所示的这个流图中，需要1024个采样值做一次FFT运算。它与滤波器模块不同，滤波器可以以sample为一个单位进行处理。



图9-17　FFT处理流图

在这个流图中，有一个“Stream to Vector”模块，它的输入类型和输出类型不同。这个模块收到1024个sample之后，输出一个长度为1024-sample的向量，整个向量是一个item。“Complex to Mag^2”这个模块把FFT模块输出的复数向量转换成幅值的平方，此时就变成了一个实数类型。所以这个模块的输入item是复数向量，输出item是实数向量。在流图中，不同类型的item端口，会用不同的颜色表示。

总之，item可以是任意类型，如一个sample、一串bit，或者是滤波器的一套系数。假如流图中的“线”是一条河流，那么其中流动的item可以是水分子，也可以是细砂，还可以是鹅卵石。

归纳：

- GNU Radio中所有的信号处理都由流图实现。
- 流图由模块组成。每个模块完成一种信号处理任务，例如滤波、信号相加、变换、解码、硬件读取信号等。

- 信号在模块间以各种格式流动，如复数类型、整型、浮点型，或者是你定义的任意类型。
- 一个流图至少要有一个source模块和一个sink模块。
- 流图程序运行时，流图将依次调用每个模块，使经过模块的信号得到处理。

9.6.3 采样率

本节讨论采样率。

在图9-16所示的例子中，音频源模块设置了固定的采样率32KS/s。因为滤波器模块没有改变信号的采样率，因此整个流图的采样率都是32KS/s。

在图9-17所示的例子中，第二个模块“Stream to Vector”把1024个输入item（浮点数）变为1个输出item（浮点型向量），可以认为item的速率变为输入速率的1/1024。这种模块可以称为“decimator”（抽取器）。如果输出速率比输入速率高，这类模块称为“interpolator”（插值器）。

在图9-17所示的例子中，流图中不同模块的速率是不同的，那么真正的基础速率是多少呢？实际上，这个例子没有真实的数据速率。只要流图中没有硬件给出一个固定的采样时钟，采样率就没有实际意义，CPU将会以尽可能快的速率处理这些信号（注意，这会导致CPU满载，像死机一样，不响应其他程序）。因此，在前面的章节中，我们提到过节流阀（Throttle）模块的重要性，它可以使整个流图模拟你设定的速率，慢速运行，避免CPU满负荷运转。

再看一个例子。在图9-18所示的这个例子中有硬件模块，就是最后的Audio Sink，这里设置的采样率是48kHz，是声卡的采样率，因此整个流图将以这个作为基础速率。这个例子还有一个特点，就是Audio Sink有两个输入端口，即左、右两个声道。两个Signal Source分别产生不同频率的声音，在声卡中合成为双音多频的拨号音。

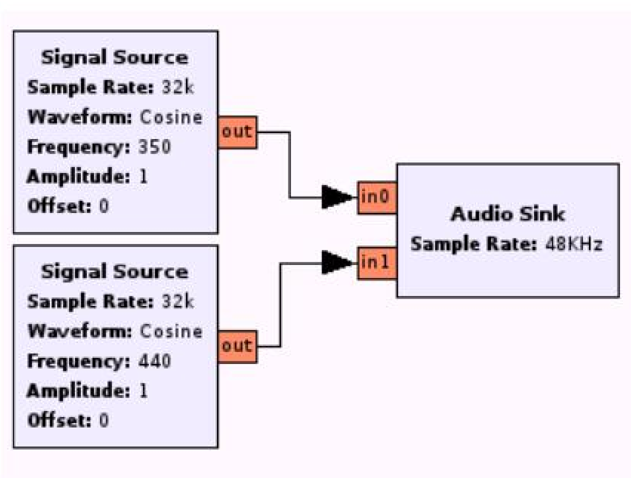


图9-18　产生拨号音的流图

9.6.4 metadata

metadata这个词不知道翻译成什么比较合适，meta前缀有“和……”、“在……之后”的含义。metadata是指伴随着信号流的一些可解析的数据标签。例如一段从USRP接收来的信号流，metadata可以包含这段信号流的接收时间、中心频率、采样率；metadata还可以带有一些特定的协议相关的信息，例如网元的标签。

在GNU Radio中，给信号流添加metadata的方法是通过一种称为“stream tag”的机制。stream tag是一个对象，跟一个item连在一起。stream tag可以是一个数值、一个向量，也可以是一个list，或者是用户定义的任何类型。

在把信号流存到文件中时，metadata也可以一起保存下来。这对离线分析信号非常有帮助，可以知道这段信号流的中心频率、采样率、接收时间等。

9.6.5 传递数据的两种方式：信号流和消息

很多模块的输入输出都是信号“流”，是无头无尾、无穷无尽的“流”。低通滤波器就是一个最典型的例子，每输入一个sample，输出端就会吐出一个滤波后的sample。滤波器不关心输入的是什么，不管是信号还是噪声，都一视同仁。

但是存在另一种数据类型—数据包（packet），例如PDU（协议数据单元），流式的处理方法就不适合了。每个数据包都是有边界的，有header，有payload。GNU Radio有两种方式处理这种数据：消息传递方式（message）和带标签的流传递（tagged stream block）。

message方式是一种异步方式，模块在内部维护一个消息队列，消息会从一个模块的队列吐出，进入另一个模块的队列。对于MAC层或者更高层的一些协议，这是一种比较适合的处理方式。一个模块可以对收到的PDU添加包头，然后把整个包作为新的PDU扔给下一个模块。

tagged stream block方式则是一串连续的、带有标签的数据块，就像一列无限长的火车，每节车厢是一个信号块，带有自己的标签。整列火车又可以看作是一个货物“流”。这种方式既可以使模块按照PDU来处理数据，也可以使模块按照流的方式来处理数据。

有些模块可以对数据在message方式和tagged stream block方式之间进行转换。

9.7 初学者如何使用GNU Radio[注]

如果你看一下GNU Radio的tutorial页面（http://gnuradio.org/redmine/projects/gnuradio/wiki/Tutorials），就会发现有太多的教程，让人眼花缭乱。本节将介绍其中最重要的部分，以节约初学者的时间，能够快速上手。

上一节我们介绍了流图的概念，下面就先介绍最简单的，使用Python编写GNU Radio应用程序。

1. 这部分内容来自http://gnuradio.org/redmine/projects/gnuradio/wiki/HowToUse。返回

9.7.1　如何编写流图—Python应用程序[注]

1.拨号音的例子

看第一个最简单的例子。下面的程序跟图9-18所示的GRC例子具有同样的功能，发出两个频率的拨号音。这个例子是gr-audio/examples/python目录下的dial_tone.py。

```
#!/usr/bin/env python

from gnuradio import gr
from gnuradio import audio
from gnuradio.eng_option import eng_option
from optparse import OptionParser

try:
    from gnuradio import analog
except ImportError:
    sys.stderr.write("Error: Program requires gr-analog.\n")
    sys.exit(1)

class my_top_block(gr.top_block):

    def _init_(self):
        gr.top_block._init_(self)

        parser = OptionParser(option_class=eng_option)
        parser.add_option("-O", "--audio-output", type="string", default="",
                          help="pcm output device name.  E.g., hw:0,0 or /dev/dsp")
        parser.add_option("-r", "--sample-rate", type="eng_float", default=48000,
                          help="set sample rate to RATE (48000)")
        (options, args) = parser.parse_args()
        if len(args) != 0:
            parser.print_help()
            raise SystemExit, 1

        sample_rate = int(options.sample_rate)
        ampl = 0.1

        src0 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 350, ampl)
        src1 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 440, ampl)
        dst = audio.sink(sample_rate, options.audio_output)
        self.connect(src0, (dst, 0))
```

```
        self.connect(src1, (dst, 1))

if _name_ == '_main_':
    try:
        my_top_block().run()
    except KeyboardInterrupt:
        pass
```

代码的第一行，告诉shell这是一个Python文件，请使用Python解释器来执行这个文件。如果要直接从命令行运行的话，文件的第一行必须这样写。

接下来的四行代码，导入了GNU Radio必需的一些Python模块。import指令，类似于C/C++中的#include，这里导入了gr、audio、eng_option和OptionParser。gr是GNU Radio中最基础的模组，所有的GNU Radio程序都必须导入这个模组。audio模组是一个与声音处理和声卡支持有关的模组。剩下的两个模组都是与参数配置和传递有关的辅助性模组。接下来try命令后面的部分，是为了导入analog模组，这个模组含有一些处理模拟信号的模块。

my_top_block这一段是这个程序的主体部分。它定义了一个名叫my_top_block的类，这个类是从gr.top_block派生出来的。top block就像一个容器，流图装在里面。top block是所有GNU Radio程序最顶层的结构。

在my_top_block这个类里面，只定义了一个成员函数def_init_(self)，它也是这个类的构造函数。这个函数的第一句代码，就是它的父类的构造函数。

```
gr.top_block._init_(self)
```

接下来的一大段用于处理输入参数，设置输入参数的类型、默认值，检查输入参数，输出帮助信息。

```
parser = OptionParser(option_class=eng_option)
        parser.add_option("-O", "--audio-output", type="string", default="",
                          help="pcm output device name.  E.g., hw:0,0 or /dev/dsp")
        parser.add_option("-r", "--sample-rate", type="eng_float", default=48000,
                          help="set sample rate to RATE (48000)")
        (options, args) = parser.parse_args()
        if len(args) != 0:
            parser.print_help()
            raise SystemExit, 1
```

参数处理后面的代码，定义了两个变量：sample_rate和ampl，分别用于设定采样率和信号的幅度。

紧接着的三行代码，产生了src0、src1和dst三个模块，也就是图9-18中的三个模块。其中src0和src1是两个信号源，它们连续地产生正弦波形，分别是350Hz和440Hz，默认采样率是48KS/s，默认幅度为0.1；dst是控制声卡的模块，把输入dst的数据用声卡播放出来。

模块的命名规则，使它们很容易理解。比如analog.sig_source_f，表示这是一个信号源，最后的f表示它输出float型的数据。因为audio_sink的输入必须是float型，而且输入值的范围需要是(-1，1)，所以信号源就选择了float型。

输入输出数据的匹配，是编程过程中必须注意的。虽然GNU Radio会自动地对数据类型做一些检查，但仍然有一些问题需要手工检查。例如，如果把整型数据发送给audio.sink，GNU Radio会报错；但如果发送了一串有些数据幅度大于1的浮点型数据流给audio.sink，它是不会报错的，它会把大于1的都当作1来处理，结果就产生了一串畸变的信号。

接下来，self.connect函数把这三个模块连接在一起。两个self.connect函数相当于画了图9-18中的两条线。connect函数的形式一般是self.connect(block1，block2，block3，…)，它会把block1的输出跟block2的输入连接，把block2的输出跟block3的输入连接，依此类推。在这个例子中，dst模块有两个输入端口，因此connect必须指明src0连接到dst的0端口，而src1连接到dst的1端口。

如果不是在GRC环境下，我们通常在拿到GNU Radio的Python程序之后，会搜索connect关键字，先把模块的连接关系搞清楚。

```
if _name_ = = '_main_':
    try:
        my_top_block().run()
    except KeyboardInterrupt:
        pass
```

代码的最后一部分，用于把top_block运行起来。top_block本身带有的run函数会初始化这个实例，然后按照构造函数的代码运行起来。

OK，运行这个程序试试吧。在终端进入gr-audio/examples/python目录，执行：

```
./dial_tone.py
```

此时你应该听到声卡发出的拨号音。如果没有听到拨号音，那一定是什么地方出错了。比如你需要指定声卡设备名称。在笔者的这个虚拟机中，需要用-O参数指定声卡设备为plughw：0，0。

```
./dial_tone.py -O plughw:0,0
```

2.利用现成的GNU Radio模块编写Python应用程序

从上一节的例子可以看到，用Python程序把GNU Radio提供的各种模块组合起来，就可以实现一些功能了。那么GNU Radio都有哪些现成的模块呢？请见表9-3。

表9-3 GNU Radio所拥有的现成的模块

模块	说明
gr	GNU Radio 的核心库。基本上必不可少
analog	模拟信号处理有关的库
audio	声卡控制的模组，有 source 和 sink 模块。可以从声卡接收声音信号，也可以把信号发送给声卡
blocks	无法分类到其他库的模块，就放在这里了
channels	仿真时用到的模拟信道的模组，例如平台衰落信道、选择性衰落信道
digital	数字信号调制解调有关的库
fec	前向纠错编码（Forward Error Correction）有关的库
fft	FFT 库
filter	滤波器
qtgui	图形界面工具库，可以用来展示时域波形、频谱、星座图、柱状图等
trellis	trellis 有关的库。trellis 是一种编码调制方式
vocoder	语音编码
wavelet	小波变换
wxgui	这个库包含了一些图形用户界面，各种窗口，窗口里的输入框、选择框、下拉选项、滑动块等

这里并没有列出所有的模组，GNU Radio的模块非常多。你可以在gnuradio根目录下，对所有的“gr-xxx”目录名快速浏览一遍，就可以大概了解现成的资源有什么了。

如果要了解模块的详细信息，可以在下面的两个页面中搜索：

http://gnuradio.org/doc/sphinx/index.html

http://gnuradio.org/doc/doxygen/index.html

这是GNU Radio自动生成的帮助文档，可以按关键字查找模块。

最后介绍一下GNU Radio模块后缀的命名规则。后缀的字母表示端口的数据类型：

f＝float

c＝complex float

i＝int

s＝short int

b＝bits（actually an integer type）

如果后缀有两个字母，就表示这个模块的输入输出分别是什么类型。比如add_const_cc，顾名思义，就是给信号加一个常量，输入是复数类型（complex），输出也是复数类型。还有三个字母后缀的，比如iir_filter_ccf，表示这是一个IIR滤波器，它的输入是复数类型，输出是复数类型，抽头系数是实数类型。

3.如何配置模块的参数

新手们常常在选择了模块之后，却不知道怎么用。在上面提到的两个帮助页面中，就可以找到所需要的信息。学会利用帮助页面，是学习GNU Radio非常重要的一步。

仍然以拨号音例子为例。看一下定义模块的这三行代码：

```
src0 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 350, ampl)
src1 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 440, ampl)
dst = audio.sink(sample_rate, options.audio_output)
```

如果想了解analog.sig_source这个模块的四个参数的含义，就可以到帮助页面去查看。以Doxygen产生的页面为例，在搜索栏中搜索“sig_source_f”，就可以找到analog.sig_source_f，如图9-19所示。

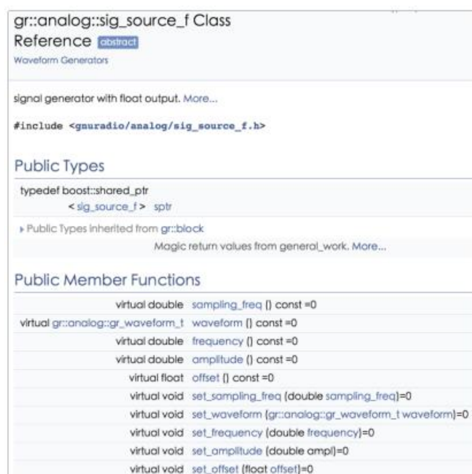


图9-19　搜索sig_source_f页面

这里给出了每个参数的类型。例如sampling_freq是double型的。offset参数还给出了默认值0。其中有个参数waveform，读者可能会问，可以选哪些波形呢？那就点击它的类型“gr_waveform_t”，可以看到有以下几种类型，如图9-20所示。

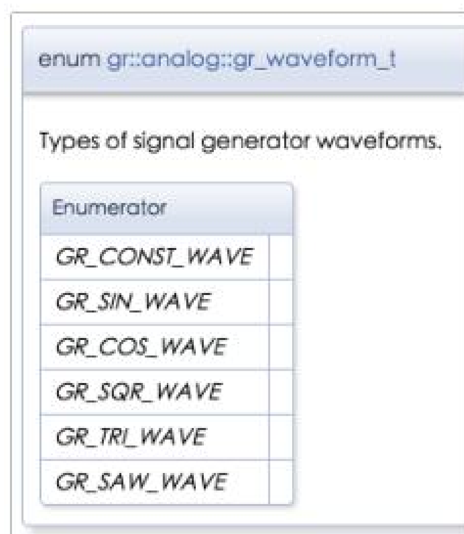


图9-20　gr_waveform_t类型

类似的audio_sink模块也可以查到它的帮助信息。在audio_sink的

帮助信息中，可以看到device_namez这个参数，有如下几种选择：

- pulse
- hw：0，0
- plughw：0，0
- surround51
- /dev/dsp

于是，通过这些帮助信息，我们就可以知道这些模块的使用方法。如果仍然有一些疑问不能解答，还有最后一招，就是直接阅读源代码。

4.用模块组合成高级模块

有时候，我们会把几个模块组合在一起，包装成一个新模块，这种模块称为“Hierarchical blocks”，译为“高级模块”。它不再是基础的、原子级模块，而是由原子组成的、类似于分子级的模块。

例如，模块B1和模块B2串联在一起，组成一个高级模块HierBlock。下面是HierBlock的伪码。

```
class HierBlock(gr.hier_block2):
    def _init_(self, audio_rate, if_rate):
        gr.hier_block2._init_(self, "HierBlock",
                       gr.io_signature(1, 1, gr.sizeof_float),
                       gr.io_signature(1, 2, gr.sizeof_gr_complex))

        B1 = blocks.block1(...) # Put in proper code here!
        B2 = blocks.block2(...)

        self.connect(self, B1, B2, self)
```

可以看到，创建一个高级模块，跟编写一个流图是非常类似的。不同之处在于，高级模块需要有_init_()部分；它作为一个模块，还需要有输入输出的定义。_init_()函数有四个参数：

- self。第一个参数必须是self。

- 高级模块的名称。在这个例子中是“HierBlock”。
- 输入端口定义。
- 输出端口定义。

gr.io_signature是用来定义输入输出端口的。它有三个参数：

- 端口个数的最小值。
- 端口个数的最大值。
- 端口的输入（输出）数据的类型。

从HierBlock这个例子来看，它的输入是float型，输出是complex型，可见数据在B1或B2中转换成了复数类型。

使用HierBlock时，跟使用其他模块的方法是一样的。

```
class FG1(gr.top_block):
    def _init_(self):
        gr.top_block._init_(self)

    ... # Sources and other blocks are defined here
    other_block1  = blocks.other_block()
    hierblock     = HierBlock()
    other_block2  = blocks.other_block()

    self.connect(other_block1, hierblock, other_block2)

    ... # Define rest of FG1
```

还可以把HierBlock的代码存到一个名为hier_block.py的文件中，然后用以下代码导入：

```
from hier_block import HierBlock
```

GNU Radio的例子中有很多这样的高级模块，比如：

```
gr-uhd/examples/python/fm_tx_2_daughterboards.py
gr-uhd/examples/python/fm_tx4.py
gr-digital/examples/narrowband/tx_voice.py
```

5.控制流图

流图都被定义成一个类，由gr.top_block派生出来的类。那么问题是，如何控制这个类的启动、运行、停止呢？

gr.top_block这个类带有一些函数，负责这些功能。所有从gr.top_block派生出来的流图也就带有同样的功能。要启动或者停止一个流图，可以使用以下这些函数，如表9-4所示。

表9-4 启动或停止一个流图的函数

函数	说明
run()	这是最简单、最常用的运行流图的方法。它实际上先调用 start()，然后调用 wait()。这个流图将会一直运行，直到它被某个条件触发而停止，或者收到 SIGINT 信号也会停止
start()	启动一个流图，让它开始运行。一旦线程创建，就返回调用者
stop()	停止一个流图的运行
wait()	等待，直到所有的模块完成任务停止运行，或者调用了 stop 指令，就会返回
lock()	锁定一个流图，准备配置参数
unlock()	解除锁定

看一个例子：

```
class my_top_block(gr.top_block):
    def _init_(self):
        gr.top_block._init_(self)
    ... # Define blocks etc. here

if _name_ = = '_main_':
    my_top_block().start()
    sleep(5) # Wait 5 secs (assuming sleep was imported!)
    my_top_block().stop()
    my_top_block().wait() # If the graph is needed to run again, wait() must be called afte
    ... # Reconfigure the graph or modify it
    my_top_block().start() # start it again
    sleep(5) # Wait 5 secs (assuming sleep was imported!)
    my_top_block().stop()  # since (assuming) the graph will not run again, no need for wa
```

在这个例子中，首先启动流图，然后等待5秒，停止这个流图；让流图进入等待状态，修改配置参数；再重启流图，等待5秒，停止。

除了以上方法，还有一些更方便的方式，用来重新配置流图，它会在设置新参数之后，自动重启流图。比如有一个乘法模块blocks.multiply_const_ff，如果查看它的文档，就会发现它有一个成

员函数blocks.multiply_const_ff.set_k()，这个函数用来设置要乘的常量。我们来看下面这个流图程序：

```
class my_top_block(gr.top_block):
    def _init_(self):
        gr.top_block._init_(self)
    ... # Define some blocks
    self.amp = blocks.multiply_const_ff(1) # Define multiplier block
    ... # Define more blocks

        self.connect(..., self.amp, ...) # Connect all blocks

    def set_volume(self, volume):
    self.amp.set_k(volume)

if _name_ == '_main_':
    my_top_block().start()
    sleep(2) # Wait 2 secs (assuming sleep was imported!)
    my_top_block.set_volume(2) # Pump up the volume (by factor 2)
    sleep(2) # Wait 2 secs (assuming sleep was imported!)
    my_top_block().stop()
```

这个流图把乘法器定义为它的一个属性self.amp，然后设置了一个set_volume()函数用来修改系数。于是在运行流图时，调用my_top_block.set_volume(2)就可以修改这个系数，然后重启这个流图。

这种方法非常常见，尤其是一些具有图形化界面的应用。比如一个频谱分析程序，我们需要不时地调整参数：中心频率、采样率、显示比例、参考功率等。当我们在输入框中输入新参数，按下回车键时，流图就以新参数重新启动了。

6.图形用户界面

如果你是Python编程方面的专家，非常擅长用Python来写GUI，那么这部分内容就可以略过了。

正如前面所介绍的，GNU Radio只不过是用一些DSP模块扩展了Python的功能。所以当你用Python写一些UI时，就可以把GNU Radio的流图添加进来。你也可以用Matplotlib或者Qwt工具，把信号可视化。

如果你只是想快速地做出一个GUI，那么GNU Radio已经为你准备了一些现成的类。这些类都是基于wxPython的，一个平台独立的GUI工具。要使用wxWidgets，你需要导入一些模组：

```
from gnuradio.wxgui import stdgui2, fftsink2, slider, form
```

上面这句代码从gnuradio.wxgui中导入了4个模块，如表9-5所示。

表9-5 导入的4个模块

模块	说明
stdgui2	基本的窗口元素
fftsink2	显示一个频谱分析仪的界面，并动态地画出 FFT 的输出结果
slider	滚动条
form	输入框

使用这些模组，我们就可以用stdgui2.std_top_block来代替gr.top_block，创建一个带有窗口的应用程序。

```
class my_gui_flow_graph(stdgui2.std_top_block):
        def _init_(self, frame, panel, vbox, argv):
                stdgui2.std_top_block._init_ (self, frame, panel, vbox, argv)
```

从上面这段代码可以看到，构造函数带有更多的参数：frame、panel、vbox、argv。stdgui2.std_top_block创建了一个窗口，带有窗口必需的一些基本元素，你可以在此基础上添加小部件（widget）。例如以下这些部件，如表9-6所示。

表9-6 小部件

部件	说明
frame	窗口。可以通过 frame.GetMenuBar()调出窗口的菜单
panel	面板。处于一个 frame 之中，可用于安置 wxControl 的部件
vbox	垂直框。用来在 panel 中对齐部件
argv	命令行参数

“form”这个模组也有很多有用的小部件：form.static_text_field()可以静态地显示文本；form.float_filed()用于输入浮点数；form.text_field()用于输入文本；form.checkbox_field()是选择框。阅读gr-wxgui/python/form.py源码可以看到所有的部件。

gnuradio.wxgui中最有用的部件，可能就是“用输入数据画图”了。因为把信号图形化，是一种最常用的分析手段。只需要从

gnuradio.wxgui中挑选一个合适的sink类的模块，连接在所需要分析的数据后面，就可以从这个sink看到信号的状况。用fftsink2来举个例子：

```
from gnuradio.wxgui import stdgui2, fftsink2

# App gets defined here ...

# FFT display (pseudo-spectrum analyzer)
my_fft = fftsink2.fft_sink_f(panel, title="FFT of some Signal", fft_size=512,
          sample_rate=sample_rate, ref_level=0, y_per_div=20)
self.connect(source_block, my_fft)
vbox.Add(my_fft.win, 1, wx.EXPAND)

if _name_ == '_main_':
  app = stdgui2.stdapp(my_gui_flow_graph, "GUI GNU Radio Application")
  app.MainLoop()
```

这里选择了fftsink2.fft_sink_f模块。它的第一个参数是panel，把panel对象传递过去。第二个参数是这个FFT图的标题“FFT of some Signal”。后面的几个参数是与FFT有关的FFT点数、采样率等。使用connect语句把source_block和这个my_fft模块连接在一起。最后把my_fft的窗口添加到vbox的框架之中。

整个流图通过app.MainLoop()运行起来。

除了gr-wxgui，还有一个gr-qtgui，是基于QT实现的GUI，也很常用。

1. 这部分内容整理自http://gnuradio.org/redmine/projects/gnuradio/wiki/TutorialsWritePythonApplications。返回

9.7.2 如何编写自己的C＋＋模块

这一节将详细地讲解如何编写自己的C＋＋模块，创建out-of-tree module。out-of-tree module就是在已有的GNU Radio代码树之外，由用户自己创建的模组，其中包含了一些模块。

在什么情况下有这样的需求呢？当你发现GNU Radio现有的模块都不能解决问题，必须亲自动手开发新模块时，就需要创建这种out-of-tree module了。

不过，在决定动手自己写模块之前，最好先到CGRAN（http://cgran.org/）网站看一下。CGRAN上有很多第三方模组，它们不属于GNU Radio代码树，而是由其他开发者编写和维护的，其中有不少经典作品。例如：请见表9-7。

表9-7 第三方模组

模组	说明
gr-osmosdr	Osmocom 项目创建的一些工具，包括 RTLSDR、HackRF、bladeRF 的 source 和 sink
gqrx	一个非常好用的频谱分析工具
gr-adsb	一台 ADS-B 接收机。在前面讲 ADS-B 安全的章节中用到过
gr-gsm	接收和分析 GSM 信号的工具
gr-lte	接收和分析 LTE 信号的工具
gr-ieee802-11	802.11 标准的收发程序

如果这些模组也不能满足你的要求，那么就自己动手吧！

创建一个新模组要用到一些工具软件。首先是gr_modtool，这是GNU Radio自带的工具，用来自动产生所有的代码模板和makefile文件，使你不必在烦琐的事情上浪费时间，可以直接关注核心的信号处理部分。gr_modtool会用最常用的设置去配置模块，因此，如果模块定制化程度很高，gr_modtool可能就不适合了。但对于阅读这本书的初学者来说，它应该是最好的选择。

除了gr_modtool，还有CMake等工具软件也是需要的。

1.创建一个out-of-tree module

module是含有多个模块的“包”，就叫模组吧，比如gr-audio、gr-digital都是module。因此，创建一个module，就会产生一个新的gr-xxx。

假设这个新的模组名字为“howto”，在所要存放代码的路径下，执行：

```
$ gr_modtool newmod howto
Creating out-of-tree module in ./gr-howto... Done.
Use 'gr_modtool add' to add a new block to this currently empty module.
```

在这个目录下，就会产生一个名为gr-howto的目录。进入这个目录，可以看到产生了很多子目录和文件。

```
$ ls
apps  cmake  CMakeLists.txt  docs  examples  grc  include  lib  python  swig
```

lib目录用于存放C＋＋/C代码文件；C＋＋/C的头文件放在include目录下；python目录当然就是存放Python文件的。

GNU Radio的C＋＋模块能够在Python下调用，是因为有SWIG（Simplified Wrapper and Interface Generator）的帮助，它是C＋＋和Python之间的桥梁。这些桥梁代码就存放在swig目录下。除非要做特别高级的修改，否则千万不要修改swig目录下的代码，gr_modtool会自动维护这部分代码。

如果你希望所创建的模块放进GNU Radio companion中使用，那么就要在grc目录下添加一些XML代码。

docs目录用来存放Doxygen和Sphinx产生的帮助文档。

apps目录里是一些完整的可执行程序，用install安装时会安装到GNU Radio的系统目录下。

examples目录可以存放一些范例程序。其他的开发者看到这些范例，就可以很快地明白该模块如何使用。

cmake就是CMake要用的目录了。另外，每个目录下都有CMakeLists.txt文件。

2.创建第一个模块square_ff

这一步我们要在gr-howto模组里面创建第一个模块，名叫square_ff。这个模块的功能是计算输入数据的平方值，从输入端口输入一个浮点数，在输出端口输出这个数的平方值，也是浮点数。所以按照GNU Radio的命名习惯，模块名称的后缀是ff。

（1）创建一个空的模块

首先，使用gr_modtool创建空文件模板，也就是先把模块的骨架搭起来，但还未在其中填充内容。在gr-howto目录下，执行：

```
gr-howto$ gr_modtool add -t general square_ff
GNU Radio module name identified: howto
Language: C++
Block/code identifier: square_ff
Enter valid argument list, including default arguments:
Add Python QA code? [Y/n]
Add C++ QA code? [y/N]
Adding file 'lib/square_ff_impl.h'...
Adding file 'lib/square_ff_impl.cc'...
Adding file 'include/howto/square_ff.h'...
Editing swig/howto_swig.i...
Adding file 'python/qa_square_ff.py'...
Editing python/CMakeLists.txt...
Adding file 'grc/howto_square_ff.xml'...
Editing grc/CMakeLists.txt...
```

根据问题的提示进行选择。在这个例子中，没有输入参数，其他问题都选择默认选项，因此一路回车，最后产生了一系列新文件。我们需要对这些文件做进一步的编程。

（2）以测试推动编程

在新模块的文件模板都生成之后，就可以编写其中的代码了。但是，强烈建议你先把测试代码写出来，以测试来推动编程，这样可以一边调试一边修改，效率更高。所以我们先写一段QA代码，打开python/qa_square_ff.py，加上以下测试代码：

```
from gnuradio import gr, gr_unittest
from gnuradio import blocks
import howto_swig as howto
```

```python
class qa_square_ff (gr_unittest.TestCase):

    def setUp (self):
        self.tb = gr.top_block ()

    def tearDown (self):
        self.tb = None

    def test_001_square_ff(self):
        src_data = (-3, 4, -5.5, 2, 3)
        expected_result = (9, 16, 30.25, 4, 9)
        src = blocks.vector_source_f(src_data)
        sqr = howto.square_ff()
        dst = blocks.vector_sink_f()
        self.tb.connect(src, sqr)
        self.tb.connect(sqr, dst)
        self.tb.run()
        result_data = dst.data()
        self.assertFloatTuplesAlmostEqual(expected_result, result_data, 6)

if _name_ == '_main_':
    gr_unittest.run(qa_square_ff, "qa_square_ff.xml")
```

gr_unittest是标准的Python模组“unittest”的一个扩展。gr_unittest添加了对浮点数和复数的支持，能够比较两个浮点数（复数）是否近似相等。Unittest会自动找到所有以“test_”开头的Python程序，并且执行。

当执行这个测试程序时，gr_unittest.main会首先触发setUp，然后执行test_001_square_ff，最后执行tearDown。

test_001_square_ff创建了一个小的流图，包含三个节点。blocks.vector_source_f(src_data)使src_data中的数据形成向量，作为向量型source；howto.square_ff是我们进行测试的模块；blocks.vector_sink_f用来收集howto.square_ff的输出。

执行run()函数运行这个流图，直到所有的测试完成。这个例子中只有一个测试函数，而有些测试包含很多个测试程序。

（3）开始编写C＋＋代码

现在我们有了一个测试例，下一步就是开始编写C＋＋代码了。gr_modtool已经产生了三个文件：lib/square_ff_impl.h、lib/square_ff_impl.cc和include/howto/square_ff.h。我们需要做的就是修改这三个文件。

首先，修改头文件。因为这个例子太简单了，所以实际上头文件完全不必修改。然后，修改lib/square_ff_impl.cc文件。gr_modtool在需要添加代码的位置，添加了gt;提示：

```
/*
 * The private constructor
 */
square_ff_impl::square_ff_impl()
  : gr::block("square_ff",
          gr::io_signature::make(<+MIN_IN+>, <+MAX_IN+>, sizeof(<+ITYPE+>
          gr::io_signature::make(<+MIN_OUT+>, <+MAX_OUT+>, sizeof(<+OTYP
{}
```

看这段代码，gt;部分都是需要修改的。square_ff的输入应该是一个浮点数，输出也是一个浮点数，所以构造函数应该写成这样：

```
/*
 * The private constructor
 */
square_ff_impl::square_ff_impl()
  : gr::block("square_ff",
          gr::io_signature::make(1, 1, sizeof(float)),
          gr::io_signature::make(1, 1, sizeof(float)))
{}
```

构造函数内部留空，因为计算平方值不需要做初始化工作。析构函数没什么要做的，保持原样。下面看一下forecast()函数：

```
void
  square_ff_impl::forecast (int noutput_items, gr_vector_int &ninput_items_required)
  {
    ninput_items_required[0] = noutput_items;
  }
```

forecast()函数的作用是，告诉调度器，要产生noutput_items输出需要多少个输入item。在这个例子中，输入输出的个数是相同的。ninput_items_required[0]里的序号0，表示这是第一个端口。如果你有兴趣，可以看一下gr：：block、gr：：sync_block、gr：：sync_decimator以及gr：：sync_interpolator的源码，其中有forecast()函数默认的示例代码。Forecast()函数是控制模块速率非常重要的工具。

最后，我们来看一下最关键的general_work()函数，这是真正进

行信号处理的部分。

```
int
  square_ff_impl::general_work (int noutput_items,
                  gr_vector_int &ninput_items,
                  gr_vector_const_void_star &input_items,
                  gr_vector_void_star &output_items)
  {
    const float *in = (const float *) input_items[0];
    float *out = (float *) output_items[0];

    // Do
    for(int i = 0; i < noutput_items; i++) {
      out[i] = in[i] * in[i];
    }

    // Tell runtime system how many input items we consumed on
    // each input stream.
    consume_each (noutput_items);

    // Tell runtime system how many output items we produced.
    return noutput_items;
  }
```

可以看到，在general_work()函数中，有一个指针指向输入缓存，另一个指针指向输出缓存，然后用一个for循环把每一个平方值写到输出缓存中。这个例子的功能非常简单，因此只需简单的几行就完成了。

（4）用CMake编译代码

在这个模组的顶层目录下，创建build目录。然后用CMake编译。

```
$ mkdir build    # We're currently in the module's top directory
$ cd build/
$ cmake ../      # Tell CMake that all its config files are one dir up
$ make           # And start building (should work after the previous section)
```

因为是在build目录下执行cmake的，这个目录就称为build tree。install tree，即安装目录，是另一个位置$prefix/lib/$pythonversion/dist-packages。其中$prefix是CMake在配置过程中设置的，默认为/usr/local/。

如果是用PyBOMBS安装GNU Radio的，那么默认的安装目录是target/。此时，要确保CMake设置的install路径是正确的。需要执行

下面的操作：

```
$ cmake -DCMAKE_INSTALL_PREFIX＝＜target_directory＞ ../ # ＜target_directory＞ sho
```

在执行完make之后，如果没有任何错误出现，在build目录下就产生了libgnuradio-howto.so等库文件。只有执行make install时，这些文件才会拷贝到install tree中。

（5）测试和调试

测试代码早就准备好了，因此直接执行make test。

```
$ make test
Running tests...
Test project /home/test/workspace/test_oot/gr-howto/build
    Start 1: test_howto
1/2 Test #1: test_howto .......................   Passed    0.01 sec
    Start 2: qa_square_ff
2/2 Test #2: qa_square_ff .....................   Passed    0.40 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   0.41 sec
```

你应该得到像上面这样的正确结果。

假如所写的代码有错误，比如把out[i]＝in[i]*in[i]；写成了out[i]＝in[i]；，没有做平方运算。那会怎样呢？显然执行make test（也可以用ctest）就会出错。

```
$ ctest
Test project /home/test/workspace/test_oot/gr-howto/build
    Start 1: test_howto
1/2 Test #1: test_howto .......................   Passed    0.00 sec
    Start 2: qa_square_ff
2/2 Test #2: qa_square_ff .....................***Failed    0.19 sec

50% tests passed, 1 tests failed out of 2

Total Test time (real) =   0.19 sec

The following tests FAILED:
   2 - qa_square_ff (Failed)
Errors while running CTest
```

显示qa_square_ff的测试没有通过。那么接下来如何调试呢？我们可以在执行ctest时加上-V和-R参数。-V表示verbose输出；-R后面的参数表示，只有匹配这个关键字的测试例才会被执行。ctest–V–R square就表示只运行qa_square_ff这个测试例，输出更多的调试信息。

```
$ ctest -V -R square
UpdateCTestConfiguration  from :/home/test/workspace/test_oot/gr-howto/build/DartC
UpdateCTestConfiguration  from :/home/test/workspace/test_oot/gr-howto/build/DartC
Test project /home/test/workspace/test_oot/gr-howto/build
Constructing a list of tests
Done constructing a list of tests
Checking test dependency graph...
Checking test dependency graph end
test 2
    Start 2: qa_square_ff

2: Test command: /bin/sh "/home/test/workspace/test_oot/gr-howto/build/python/qa_s
2: Test timeout computed to be: 9.99988e+06
2: F
2: ======================================================================
2: FAIL: test_001_square_ff (__main__.qa_square_ff)
2: ----------------------------------------------------------------------
2: Traceback (most recent call last):
2:   File "/home/test/workspace/test_oot/gr-howto/python/qa_square_ff.py", line 44, in t
2:     self.assertFloatTuplesAlmostEqual(expected_result, result_data, 6)
2:   File "/usr/local/lib/python2.7/dist-packages/gnuradio/gr_unittest.py", line 90, in ass
2:     self.assertAlmostEqual (a[i], b[i], places, msg)
2: AssertionError: 9 != -3.0 within 6 places
2:
2: ----------------------------------------------------------------------
2: Ran 1 test in 0.002s
2:
2: FAILED (failures=1)
1/1 Test #2: qa_square_ff .....................***Failed    0.16 sec

0% tests passed, 1 tests failed out of 1

Total Test time (real) =   0.17 sec

The following tests FAILED:
   2 - qa_square_ff (Failed)
Errors while running CTest
```

从输出的信息可以看到，有9！=-3.0错误。于是，我们就可以定位到运算结果是错误的。注意，这里的调试是Python语言这个层次的，如果要调试C代码，需要用其他的方法。

3.创建第二个模块square2_ff

通常一个模组中含有很多个模块，现在我们在模组中添加第二个模块，名为square2_ff。它的功能跟第一个模块square_ff一样，都是计算平方值。不同的是，我们将以gr：：sync_block为基础，派生出这个模块。

gr：：sync_block是一个从gr：：block派生出来的子类，“sync”表示它是一个同步的模块，具体来说，就是它的输入输出速率是1：1。如果速率比不是1：1，则还有两个子类，名为sync_decimator和sync_interpolator，它们是从sync_block派生出来的，分别用于降低速率（抽取）和升高速率（插值）。

因为sync_block设定了输入输出速率相等，所以它会省略ninput_items参数，还会省略consume_each()函数。

按照好习惯，先来准备测试代码。修改qa_square_ff.py，添加一个新的测试函数。这个测试函数除了名字不一样，内容跟test_001_square_ff是完全一样的。

```
def test_002_square2_ff(self):
    src_data = (-3, 4, -5.5, 2, 3)
    expected_result = (9, 16, 30.25, 4, 9)
    src = blocks.vector_source_f(src_data)
    sqr = howto.square2_ff()
    dst = blocks.vector_sink_f()
    self.tb.connect(src, sqr, dst)
    self.tb.run()
    result_data = dst.data()
    self.assertFloatTuplesAlmostEqual(expected_result, result_data, 6)
```

然后，使用gr_modtool产生模板文件。不必产生QA代码，因为我们已经添加到qa_square_ff.py里了。

```
$ gr_modtool add -t sync square2_ff
GNU Radio module name identified: howto
Language: C++
Block/code identifier: square2_ff
Enter valid argument list, including default arguments:
Add Python QA code? [Y/n] n
Add C++ QA code? [Y/n] n
Adding file 'lib/square2_ff_impl.h'...
Adding file 'lib/square2_ff_impl.cc'...
```

```
Adding file 'include/howto/square2_ff.h'...
Editing swig/howto_swig.i...
Adding file 'grc/howto_square2_ff.xml'...
Editing grc/CMakeLists.txt...
```

此时，打开square2_ff_impl.cc，就会发现它与square_ff_impl.cc略有不同，因为它的父类是gr：：sync_block。我们把需要添加的代码写进去。

```
/*
  * The private constructor
  */
 square2_ff_impl::square2_ff_impl()
  : gr::sync_block("square2_ff",
       gr::io_signature::make(1, 1, sizeof(float)),
       gr::io_signature::make(1, 1, sizeof(float)))
 {}

 /*
  * Our virtual destructor.
  */
 square2_ff_impl::~square2_ff_impl()
 {
 }

 int
 square2_ff_impl::work(int noutput_items,
  gr_vector_const_void_star &input_items,
  gr_vector_void_star &output_items)
 {
   const float *in = (const float *) input_items[0];
   float *out = (float *) output_items[0];

   // Do
   for(int i = 0; i < noutput_items; i++) {
     out[i] = in[i] * in[i];
   }

   // Tell runtime system how many output items we produced.
   return noutput_items;
 }
```

square2_ff_impl：：work的输入参数中没有gr_vector_intamp;ninput_items，而且没有forecast()函数。

特别要介绍的一点是，如果一个模块在处理数据时，必须要历史数据协助的话，那么可以用set_history()函数设置历史数据的数量。

在滤波器类的模块中，有很多这样的例子，比如当前时刻T的输出，就需要用到T-1至T-N的数据。例如在pfb_arb_resampler_ccc_impl.cc中，有一行是：

```
set_history(d_resamp->taps_per_filter());
```

意思是在多相数字滤波器中，历史数据的数量至少要等于滤波器的抽头数量。

```
set_history(d_resamp->taps_per_filter());
```

修改完代码之后，执行：

```
$ cmake ../
$ make
$ ctest
```

如果测试通过，第二个模块就成功创建了。

4.把模块添加到GRC图形环境下

如果执行make install命令，新创建的模块就会被拷贝到安装目录下，但是不会出现在GRC中。如果想在GRC中使用新模块，就需要用gr_modtool添加XML代码。

```
$ gr_modtool makexml square2_ff
GNU Radio module name identified: howto
Warning: This is an experimental feature. Don't expect any magic.
Searching for matching files in lib/:
Making GRC bindings for lib/square2_ff_impl.cc...
Overwrite existing GRC file? [y/N] y
```

在大部分情况下，gr_modtool只是创建了代码模板，还需要对XML代码进行修改。但是，我们现在这个例子实在是太简单了，所以不需要修改就可以用了。看一下grc/howto_square2_ff.xml文件。

```
<block>
  <name>Square2 ff</name>
  <key>howto_square2_ff</key>
  <category>HOWTO</category>
```

```
  <import>import howto</import>
  <make>howto.square2_ff()</make>
  <sink>
    <name>in</name>
    <type>float</type>
  </sink>
  <source>
    <name>out</name>
    <type>float</type>
  </source>
</block>
```

然后，就可以把它安装到GRC中了。执行：

```
$ sudo make install
$ sudo ldconfig
```

于是，我们就可以在GRC中找到它了。如果GRC处于正在运行的状态，点击“Reload”按钮，就可以把新安装的模块加载进来。可以看到，在右侧模块列表中，有一个“HOWTO”模组，其中有一个模块叫“square2ff”。

到此为止，如何编写自己的C++模块就介绍完了。

最后总结一下创建一个新模组的步骤。

（1）创建模组：gr_modtool create MODULENAME

（2）向模组中添加模块：gr_modtool add BLOCKNAME

（3）创建编译目录：mkdir build/

（4）编译：cd build && cmake <OPTIONS> ../&& make

（5）测试：make test或ctest，或者ctest–V

（6）安装：sudo make install

（7）Ubuntu用户需要重新加载库文件：sudo ldconfig

9.7.3　如何编写自己的Python模块

上一节介绍了怎样编写C＋＋模块，模块也可以用Python来写。在9.7.1节中，讲了一些有关的方法，用模块组成高级模块。与之不同的是，在此我们要介绍用Python语言来编写一个模块。

Python是一种解释型语言，所以它的运行效率是比较低的。因此在设计这个模块时，要考虑到用Python是否能够满足计算性能的要求。

我们要在之前创建的howto模组中添加一个用Python写的square3_ff模块。首先，编写测试代码，继续在qa_square_ff.py中添加测试例。

```
from gnuradio import gr, gr_unittest
from gnuradio import blocks
import howto_swig
from square3_ff import square3_ff

class qa_square_ff (gr_unittest.TestCase):

    def setUp (self):
        self.tb = gr.top_block ()

    def tearDown (self):
        self.tb = None

    # [...] Skipped the other test cases

    def test_003_square3_ff (self):
        src_data = (-3, 4, -5.5, 2, 3)
        expected_result = (9, 16, 30.25, 4, 9)
        src = blocks.vector_source_f (src_data)
        sqr = square3_ff ()
        dst = blocks.vector_sink_f ()
        self.tb.connect (src, sqr)
        self.tb.connect (sqr, dst)
        self.tb.run ()
        result_data = dst.data ()
        self.assertFloatTuplesAlmostEqual (expected_result, result_data, 6)

if _name_ == '_main_':
```

```
    gr_unittest.main ()
```

这段测试代码跟前面的两段几乎是一样的。不同的是，在最前面的import部分，需要导入这个Python模块，否则Python程序会找不到square3_ff。

```
from square3_ff import square3_ff
```

然后，编写模块代码。使用gr_modtool工具添加这个模块。

```
$ gr_modtool add -t sync -l python square3_ff
GNU Radio module name identified: howto
Language: Python
Block/code identifier: square3_ff
Enter valid argument list, including default arguments:
Add Python QA code? [Y/n] n
Adding file 'python/square3_ff.py'...
Adding file 'grc/howto_square3_ff.xml'...
Editing grc/CMakeLists.txt...
```

接下来，修改python/square3_ff.py文件，改完之后应该是下面这样的：

```
import numpy
from gnuradio import gr

class square3_ff(gr.sync_block):

    def _init_ (self):
        gr.sync_block. _init_ (self,
            name="square3_ff",
            in_sig=[numpy.float32],
            out_sig=[numpy.float32])

    def work(self, input_items, output_items):
        in0 = input_items[0]
        out = output_items[0]
        # <+signal processing here+>
        out[:] = in0 * in0
        return len(output_items[0])
```

跟C++模块对比，可以看到它也有构造函数（_init_()）；也要规定输入输出类型，都是numpy.float32；也有一个work()函数。

如果这个模块有多个端口，那么用Python语言描述应该是这样的：[(numpy.float32，4)，numpy.float32]。这表示输入（或输出）有两个端口，其中第一个端口是向量型的，是由4个浮点数组成的向量；第二个端口是一个浮点数。

在输出部分，向output_items赋值时，一定要使用[：]操作符，保证数据拷贝正确。

我们来测试这个模块。

```
$ ctest
Test project /home/test/workspace/test_oot/gr-howto/build
    Start 1: test_howto
1/2 Test #1: test_howto .......................   Passed    0.00 sec
    Start 2: qa_square_ff
2/2 Test #2: qa_square_ff .....................   Passed    0.17 sec

100% tests passed, 0 tests failed out of 2
```

没有问题。

与C++模块类似，Python模块也有多种类型：

- gr.sync_block
- gr.decim_block
- gr.interp_block
- gr.basic_block

也有forecast()、work()、general_work()函数，它们的输入参数要写成下面这样的形式：

```
def work(self, input_items, output_items):
    # Do stuff

    def general_work(self, input_items, output_items):
    # Do stuff
```

通过len(input_items[PORT_NUM])可以获得输入数据的个数。

到此为止，这部分内容就介绍完了。如果你是一位初学者，相信

你已经掌握了基本的GNU Radio编程方法。

9.7.4 调试代码的方法

前面已经介绍了一些用Python编写的QA代码进行调试的方法。这里要介绍一些其他的调试方法。首先介绍几种比较简单，但是非常有用的方法。

1.使用GRC中的调试模块

如果你是用GRC来开发应用的，那么直接使用GRC中的一些模块进行调试是很方便的。最常用的模块是Oscilloscopes sink、number sink和FFT sink，把这些sink模块连接到你要观察的模块的输出或输入端口，就可以观察到波形、数值或者频谱。你可以很直观地看到数据流的内容，从而判断出这个模块是否有错误。

当不需要调试时，可以禁用这个仅仅用来观察的sink模块。如果彻底不需要它，则可以删掉。

2.把中间结果转存到文件中

如果要对模块之间的数据流做细致分析，则需要把中间的数据流保存到文件中，然后用其他工具软件做离线分析。常用的工具软件有GNU Radio、Octave、SciPy，还有MATLAB。

转存数据最简单的办法，就是连接到一个file sink模块，它会把经过这里的全部数据流存到指定的文件中。

当要调试C＋＋模块内部的代码时，特别是当你不能通过QA代码和打印输出来定位错误时，就不得不用到单步调试工具了。

3.用gdb调试C＋＋模块

在Python代码中，在import之后，打印出进程id，并等待键盘输

入。此时打开另一个窗口，运行gdb，告诉它进程id，使它附着在这个Python进程上。然后，就可以在任意位置设置断点进行调试了。结束调试时，回到Python程序窗口，点击任意键就可以继续运行。

我们继续用前面的square2_ff的例子来尝试调试。先写一个简单的GNU Radio应用程序。

```
""" Testing GDB, yay """

import os
from gnuradio import gr
import howto

class SquareThat(gr.top_block):
    def _init_(self):
        gr.top_block._init_(self, name="square_that")
        self.src = gr.vector_source_f((1, 2, 3, 4, )*5)
        self.sqr = howto.square2_ff()
        self.sink = gr.vector_sink_f()
        self.connect(self.src, self.sqr, self.sink)

def main():
    """ go, go, go """
    top_block = SquareThat()
    top_block.run()

if _name_ == "_main_":
    print 'Blocked waiting for GDB attach (pid = %d)' % (os.getpid(),)
    raw_input ('Press Enter to continue: ')
    main()
```

然后对howto模组用debug模式重新编译。

```
$ cmake .. –DCMAKE_BUILD_TYPE=Debug
$ make
$ sudo make install
```

接下来就可以调试这个程序了。假设应用程序的文件名为test_gdb.py。

```
$ python test_gdb.py
Blocked waiting for GDB attach (pid = 27049)
Press Enter to continue:
```

此时程序就停下来，等待键盘输入之后再继续。于是，我们打开

另一个终端窗口。

```
$ sudo gdb –p 27049
```

注意，Ubuntu用户要加sudo才有权限调试。此时，gdb会停在Python程序暂停的地方。

```
Loaded symbols for /usr/local/lib/libgnuradio-howto.so
0xb7755424 in _kernel_vsyscall ()
(gdb)
```

然后设置断点，把断点设置在work()函数这里。你可以双击Tab键自动填充关键字。设好断点之后，输入c，继续运行这个程序。

```
(gdb) break gr::howto::square2_ff_impl::work(int, std::vector<void const*, std::
allocator<void const*> >&, std::vector<void*, std::allocator<void*> >&)
Breakpoint 1 at 0xb5620d8a: file /home/test/workspace/test_oot/gr-howto/lib/square2_
(gdb) c
Continuing.
```

接着在Python程序窗口，从键盘任意输入一个字符，使程序继续运行。这时，在gdb调试窗口就可以看到有调试信息输出，程序会停在work()函数入口的位置。接下来你就可以用gdb的各种命令进行调试了。我们可以单步执行，打印一些变量的值。程序继续运行，最后终止。退出gdb调试。

```
(gdb) c
Continuing.
[New Thread 0xb3d3bb40 (LWP 10827)]
[Thread 0xb3d3bb40 (LWP 10827) exited]
[New Thread 0xb353ab40 (LWP 10828)]
[Switching to Thread 0xb353ab40 (LWP 10828)]

Breakpoint 1, gr::howto::square2_ff_impl::work (this=0x8604680,
    noutput_items=20, input_items=..., output_items=...)
    at /home/test/workspace/test_oot/gr-howto/lib/square2_ff_impl.cc:60
60        const float *in = (const float *) input_items[0];
(gdb) n
61        float *out = (float *) output_items[0];
(gdb)
64        for(int i = 0; i < noutput_items; i++) {
(gdb)
65          out[i] = in[i] * in[i];
(gdb) n
```

```
64        for(int i = 0; i < noutput_items; i++) {
(gdb) n
65          out[i] = in[i] * in[i];
(gdb) print in[i]
$6 = 2
(gdb) print out[i]
$7 = 0
(gdb) n
64        for(int i = 0; i < noutput_items; i++) {
(gdb) print out[i]
$8 = 4
(gdb) c
Continuing.
[New Thread 0xb2bffb40 (LWP 10829)]
[Thread 0xb2bffb40 (LWP 10829) exited]
[New Thread 0xb23feb40 (LWP 10835)]
[Thread 0xb353ab40 (LWP 10828) exited]
[Thread 0xb23feb40 (LWP 10835) exited]
[Inferior 1 (process 10820) exited normally]
(gdb) quit
```

OK，通过上面这样的方法，就可以调试C++模块了。

还有一种调试方法是利用core dump文件。有时候会出现这样的情况，程序在正常运行时出错，可是在用gdb单步执行时却没有出错。这是因为gdb单步调试速度太慢，因此问题没有机会暴露出来。此时，就可以用core dump的方式来分析错误。

首先需要启用core dump（在默认情况下它是禁用的）。使用ulimit–a命令查看当前状态。

```
$ ulimit -a
core file size          (blocks, -c) 0
```

我们把这个参数设为无限大。

```
$ ulimit -c unlimited
$ ulimit -a
core file size          (blocks, -c) unlimited
```

然后运行应用程序，当程序异常中止时，就会在当前目录下产生一个名为core或者core.PID的文件。假设core文件的名字是core.12345，把它加载到gdb。

```
gdb /usr/bin/python core.12345
```

然后执行：

```
i stack
```

显示程序执行的过程。

在软件无线电这样的用途中，我们写的程序往往都是连接了硬件，做实时信号处理，单步调试一般是不可能的。所以常用的调试方法主要是用GRC中的sink模块、中间数据转存和core dump文件。

9.8 范例解读—OFDM Tunnel

这一节将详细讲解GNU Radio自带的一个例子—OFDM Tunnel。这部分内容是2010年写的，现在（2016年）的代码与当年的相比略有出入，但大体结构是一致的，不影响阅读和理解。

Tunnel是GNU Radio中很经典的例子。Tunnel有两个，其中一个是基于GMSK调制的（gr-digital/examples/narrowband）；另一个基于OFDM调制的（gr-digital/examples/ofdm）。它们都由物理层和MAC层构成，提供一个虚拟的Ethernet接口，使得基于IP的各种应用程序都可以加载在这个Tunnel上面，它就像一个管道，负责传输数据。

首先我们来运行一下GMSK Tunnel[注]（这个例子运行成功的概率很高，OFDM Tunnel有时会由于某些原因不灵）。

（1）用两个USRP分别加上子板，例如我们用了两个RFX900，然后连接到两台PC上。

（2）在PC_A上，打开两个终端窗口：在第一个终端窗口中，输入下列命令：

```
$./tunnel.py --freq 930M --bitrate 500k –v
```

它告诉程序在930MHz上建立一个速率为500kbps的连接，并把每个数据包的一些信息输出到屏幕。在第二个终端窗口中，输入：

```
$ ifconfig gr0 192.168.200.1
```

它的意思是给接口gr0配置一个IP地址192.168.200.1。

（3）在PC_B上，打开两个终端窗口：在第一个终端窗口中，输入：

```
$./tunnel.py --freq 930M --bitrate 500k –v
```

这样，PC_B就会与PC_A一样，在930MHz上有一个500kbps的物理层链路，两者才能够相互通信。在第二个终端窗口中，输入：

```
$ ifconfig gr0 192.168.200.2
```

它给PC_B的gr0配置了另一个IP地址192.168.200.2，以区别于PC_A。

OK，程序运行起来之后，在PC_A的第一个终端窗口中，我们就会看到以下这些输出：

```
# ./tunnel.py --freq 930M --bitrate 500k -v
>>> gr_fir_fff: using SSE
bits per symbol = 1
Gaussian filter bt = 0.35
Tx amplitude 0.25
modulation: gmsk_mod
bitrate: 500kb/s
samples/symbol: 2
USRP Sink: A: Basic Tx
Requested TX Bitrate: 500k Actual Bitrate: 500k
bits per symbol = 1
M&M clock recovery omega = 2.000000
M&M clock recovery gain mu = 0.175000
M&M clock recovery mu = 0.500000
M&M clock recovery omega rel. limit = 0.005000
frequency error = 0.000000
Receive Path:
modulation: gmsk_demod
bitrate: 500kb/s
samples/symbol: 2
USRP Source: A: Basic Rx
Requested RX Bitrate: 500k
Actual Bitrate: 500k
modulation: gmsk
freq: 930M
bitrate: 500kb/sec
samples/symbol: 2
Carrier sense threshold: 30 dB

Allocated virtual ethernet interface: gr0
You must now use ifconfig to set its IP address. E.g.,

$ sudo ifconfig gr0 192.168.200.1

Be sure to use a different address in the same subnet for each machine.
```

```
Tx: len(payload) = 90
Tx: len(payload) = 54
Tx: len(payload) = 153
Tx: len(payload) = 82
Tx: len(payload) = 235
Rx: ok = False len(payload) = 235
Tx: len(payload) = 78
Tx: len(payload) = 235
```

最后这部分信息说明了PC_A发送了几个数据包，长度分别都说明了；收到了一个长度为235的数据包，但是校验结果是错误的。

1. 运行GMSK Tunnel这部分内容来自Alalelddin Mohammed的硕士论文Studying Media Access and Control Protocols。返回

9.8.1 系统框图和MAC帧的构成

如图9-21所示是Tunnel的系统框图。Tunnel的物理层由发射机、接收机和一个载波侦听（sensing probe）三部分构成，完成由信息比特到基带波形的转换，以及通过能量检测判断当前信道是否空闲。MAC层是一个基于CSMA（Carrier Sense Multiple Access）的简单的MAC层。MAC层与PHY层之间传递的是一个在IP包的基础上加了一些包头和包尾的数据包。

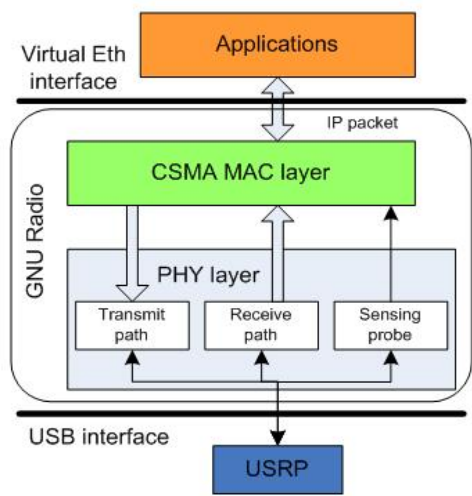


图9-21　Tunnel的系统框图

图9-22说明了一个IP数据包是如何打包成MAC数据包的。

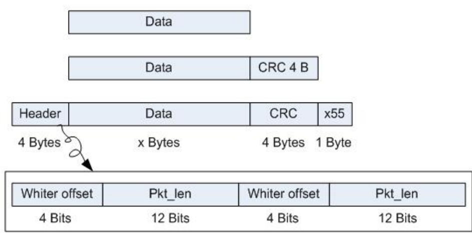


图9-22　IP数据包到MAC数据包的打包过程

首先，IP数据包被加上了4字节的校验比特，算法是CRC32。然后，数据部分加上CRC比特和尾比特（x55），都被白化处理，使得数据具有随机均匀分布的特性。最后，加上一个4字节的包头。包头包含两个信息：白化参数4比特和数据包长度12比特。包头采用了重复发送的方法，以增加可靠性。至此，一个完整的MAC数据包就包装完成了。

9.8.2　物理层

下面以OFDM Tunnel为例来解读物理层。

先来看OFDM Tunnel发射机框图，如图9-23所示。

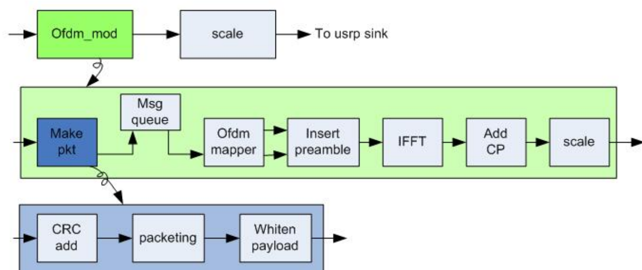


图9-23　OFDM Tunnel发射机框图

在transmit_path.py中，语句：

```
self.connect(self.ofdm_tx, self.amp, self)
```

说明其中包含两个模块：ofdm_tx是一个ofdm_mod类；amp是一个乘法器。

进入ofdm_mod类看一下，其代码在ofdm.py文件中。

在ofdm_mod中，数据包首先经过一个send_pkt()函数，完成MAC包的打包过程。

```
send_pkt(self, payload='', eof=False)
```

然后MAC包被放进一个队列中。

```
self._pkt_input.msgq().insert_tail(msg)
```

后面的ofdm_mapper_bcv模块从队列中取出数据包，根据OFDM调制的参数映射成一个个OFDM符号，再发送到后续模块，添加前导序列（Preamble），进行IFFT变换，添加循环前缀（Cyclic Prefix，

CP），最后调整一下幅度，发送出去。

这里想特别提一下的是，在ofdm_mapper之后是流图的形式，在这之前是通过一个消息队列（message queue）与MAC层联系在一起的。这种连接方式使得“异步的”MAC层数据（而且数据包长不定），跟与系统时钟“同步的”物理层连接在一起。这种连接方式是一个很好的例子，值得参考。

接收机框图见图9-24。

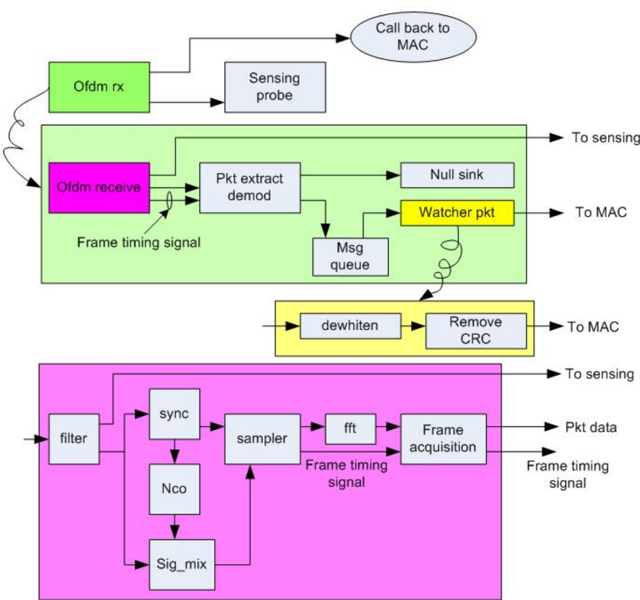


图9-24　OFDM Tunnel接收机框图

receive_path.py包含了ofdm_demod和probe两个模块。ofdm_demod显然就是OFDM接收机部分；而probe是一个信号检测模块，当USRP收到的信号幅度大于门限时，就认为无线信道已经被其他用户占用。

ofdm_demod类的代码在ofdm.py文件中，主要分成同步模块（ofdm_receiver）、解调模块（ofdm_frame_sink）和MAC帧拆包部分。与发射部分类似，物理层与MAC层也是通过一个队列self._rcvd_pktq连接在一起的。ofdm_receiver部分比较复杂，是用Python写的，完成了帧同步、频偏估计、频偏纠正、FFT的功能。ofdm_frame_sink是一个用C写成的模块，完成了从调制符号到比特的

解映射过程（不确定有没有信道均衡）。

9.8.3 调试方法

整个OFDM Tunnel的物理层还是比较简单的，它模仿了802.11的物理层，在不定长的突发数据包前面添加一个定长的前导序列，依靠这个前导序列完成时间同步和频率同步。但它没有信道编码，因此抗噪声性能较差。

在gnuradio-examples\python\ofdm目录下，除了Tunnel调用的函数外，还有许多其他的函数。这些函数都是在程序开发过程中需要用到的，它们教会了我们如何一步步地进行程序开发。特别是对于利用GNU Radio做物理研究的人员来说，是很好的参考。下面简单说明一下。

- ofdm_mod_demod_test.py—用于物理层收发模块的仿真测试。

- benchmark_ofdm.py—加上MAC层以后，做收发的仿真测试。

- benchmark_ofdm_tx.py、benchmark_ofdm_rx.py—加上USRP之后，做单向收发的测试。分别测试了连续的数据包传输和不连续的突发数据包传输。

当单向传输没有问题之后，就可以实验双向传输了：tunnel.py。

另外，还有一些MATLAB程序，帮助调试程序。当我们把log标志设为True时，就会产生很多.dat文件。看下面这段代码：

```
if logging:
        self.connect(self.chan_filt,gr.file_sink(gr.sizeof_gr_complex, "ofdm_receiver-chan_fi
        self.connect(self.fft_demod,gr.file_sink(gr.sizeof_gr_complex*fft_length, "ofdm_rece
        self.connect(self.ofdm_frame_acq,gr.file_sink(gr.sizeof_gr_complex*occupied_tones
        self.connect((self.ofdm_frame_acq,1), gr.file_sink(1, "ofdm_receiver-found_ corr_b.d
        self.connect(self.sampler,gr.file_sink(gr.sizeof_gr_complex*fft_length, "ofdm_receiv
        self.connect(self.sigmix, gr.file_sink(gr.sizeof_gr_complex, "ofdm_receiver- sigmix_
        self.connect(self.nco, gr.file_sink(gr.sizeof_gr_complex, "ofdm_receiver- nco_c.dat")
```

这些文件把各个模块的输出都记录下来：同步之前、频率同步之后、FFT之后、解映射之后等。然后用MATLAB程序进行一一检查，

就可以发现究竟哪一步出了问题。也就是前面所介绍的，把中间结果转存到文件中。

总结这个例子的开发方法，我们建议：如果你要使用GNU Radio创建自己的一个无线传输系统，请按如下步骤进行。

第一步：用MATLAB写一个物理层收发程序，设计各个功能模块，确定参数等。

第二步：用GNU Radio写一个不包括USRP的收发程序，与MATLAB程序一致，方便把GNU Radio中的数据导入MATLAB中调试。

第三步：当物理层没有问题之后，再添加MAC层。

第四步：加入USRP。先调试单向通信，再调试双向通信。